Πανεπιστήμιο Θεσσαλίας

UNIVERSITY OF THESSALY

Διπλωματική Εργασία του Προπτυχιακού Φοιτητή Μπάρδα Γεώργιου

"Γρήγορη Ανίχνευση Κινούμενων Αντικειμένων για Εφαρμογές Ψηφιακού Βίντεο"

" Fast Detection of Moving Objects for Digital Video Applications"

Επιβλέποντες Καθηγητές:
Κατσαβουνίδης Ιωάννης
Αντωνόπουλος Χρήστος

Τμήμα Μηχανικών Η/Υ και Δικτύων/ΗΜΜΥ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Σεπτέμβρης/Οκτώβρης 2013

gebardas@uth.gr

Πρόλογος

Στις διάφορες εφαρμογές βίντεο, συμπεριλαμβανομένης της συμπίεσης βίντεο, χρησιμοποιούνται διάφοροι αλγόριθμοι που βρίσκουν δύο μεγέθη τα οποία μπορούν να μας δώσουν πολλές πληροφορίες για τη φύση, τον τύπο αλλά και την κίνηση των αντικειμένων του βίντεο. Αυτά τα δύο μεγέθη είναι τα διανύσματα κίνησης (motion vectors) και η διασπορά των υπολειπόμενων τιμών των pixels (pixel residual variance) του κάθε κομματιού του βίντεο, γνωστό και ως macroblock (συνήθως, κομμάτι μεγέθους 16x16 pixels). Και δεν είναι μόνο η πληροφορία που δίνουν που τα καθιστά τόσο σημαντικά, αλλά και η ευκολία να τα εξάγουμε από το βίντεο, μιας και τα διανύσματα κίνησης είναι πληροφορία που εξάγεται και χρησιμοποιείται άμεσα κατά τη διαδικασία συμπίεσης βίντεο ενώ η διασπορά υπολειπόμενων τιμών των pixels εξάγεται σχετικά εύκολα, μετά από μικρή επεξεργασία των pixels του macroblock. Μέσα από αυτά τα δύο μεγέθη, στα οποία βασίζεται κατά κύριο λόγο η ανάπτυξη της συγκεκριμένης εργασίας, θα δούμε τρόπους και τεχνικές με τις οποίες θα πάρουμε σοβαρές πληροφορίες για το βίντεο μας και τις κινήσεις που παρατηρούνται μέσα σε αυτό. Τελικός σκοπός μας είναι μέσα από τις παραπάνω πληροφορίες να μπορούμε να καταλάβουμε και να επισημάνουμε τα διαφορετικά αντικείμενα του βίντεο, λαμβάνοντας υπόψη της παραμέτρους κίνησης. Αυτή η νέα πληροφορία θα μας ανοίξει και νέες προοπτικές στον τομέα της επεξεργασίας βίντεο καθότι μπορούμε να τη χρησιμοποιήσουμε σε πάρα πολλές εφαρμογές, όπως βιντεοπαρακολούθηση (video surveillance), ασφάλεια κτηρίων και υποδομών, ασφάλεια κατά την οδήγηση, και άλλα.

gebardas@uth.gr

Κατάλογος περιεχομένων

Κεφάλαιο 1ο:Εισαγωγή

Περιγραφή του προβλήματος

Η ανάγκη αναγνώρισης κινούμενων αντικειμένων σε εφαρμογές βίντεο υπάρχει ανέκαθεν, από την πρώτη στιγμή που ανακαλύφθηκε το βίντεο σαν έννοια. Από την αρχή όμως υπήρχε και η αντίληψη πως το βίντεο δεν μπορεί να δώσει πληροφορία μέσω των τιμών φωτεινότητας και χρωματικότητας (luminocity & chromaticity) πέραν της συνολικής εικόνας του μέσα από τα διάφορα χρώματα των pixels, γι αυτό και οι πρώτες λογικές αναγνώρισης αντικειμένων στηρίχθηκαν σε πολύπλοκους τρόπους συλλογής πληροφορίας για τα αντικείμενα, κυρίως βασισμένους σε τεχνικές επεξεργασίας (στατικής) εικόνας, όπως αναγνώριση περιγράμματος, περιοχών με συγκεκριμένες αποχρώσεις, όπως το χρώμα του δέρματος, κλπ. Η συμπίεση όμως του βίντεο ήταν αυτή που έφερε στο προσκήνιο δύο σημαντικά νέα μεγέθη για την επίλυση αυτού του προβλήματος:

- **Διάνυσμα κίνησης (Motion vector - mv)**: εν ολίγοις είναι ένα διάνυσμα δύο τιμών, στη μορφή (mvx,mvy) που μας λέει ποια είναι η θέση ενός macroblock σε σχέση με τη θέση του στο προηγούμενο καρέ. Δηλαδή μπορούμε να πούμε ότι το mv είναι το διάνυσμα που δείχνει την κίνηση που έκανε το συγκεκριμένο macroblock ανάμεσα στα δύο διαδοχικά καρέ του βίντεο.
- **Διασπορά υπολειπόμενων pixels (Pixel residual variance)**: Υπολειπόμενα pixels (pixel residuals) ονομάζονται οι τιμές των διαφορών των αρχικών pixels μιας εικόνας – ή ενός κομματιού της εικόνας, π.χ. ενός macroblock – και των pixels αναφοράς, όπως αυτά προκύπτουν από τη διαδικασία εύρεσης διανύσματος κίνησης που αναφέρθηκε παραπάνω. Για παράδειγμα, αν μέσω του διανύσματος κίνησης καταφέρουμε να βρούμε ακριβώς τις ίδιες τιμές των pixels του τρέχοντος καρέ, σε κάποια άλλη θέση στο προηγούμενο καρέ, τότε τα υπολειπόμενα pixels θα έχουν όλα τιμή ίση με μηδέν. Αυτό είναι μια σαφής ένδειξη ότι το εν λόγω macroblock μετακινήθηκε ολόκληρο κατά το μέγεθος του αντίστοιχου διανύσματος κίνησης. Αν, όμως, η διαδικασία εύρεσης κίνησης καταλήξει σε μια θέση στο προηγούμενο καρέ, η οποία μοιάζει πολύ λίγο το τρέχον καρέ, όπως για παράδειγμα συμβαίνει για πρωτοεμφανιζόμενα αντικείμενα στο βίντεο, τότε τα υπολειπόμενα pixels αναμένεται να έχουν μεγάλες τιμές, τόσο θετικές όσο και αρνητικές. Μια σημαντική μετρική είναι η διασπορά των εν λόγω τιμών, αφού στην περίπτωση που έχουμε ένα αντικείμενο το οποίο ανήκει *μερικώς* σε κάποιο macroblock, αναμένουμε κάποια από τα pixels να έχουνε μικρές υπολειπόμενες τιμές ενώ άλλα pixels θα έχουν μεγάλες (κατ'απόλυτη τιμή) τιμές. Όπως γνωρίζουμε, η διασπορά *n* τιμών μας λέει πόσο κοντά είναι αυτές οι τιμές μεταξύ τους ή καλύτερα, πόσο απέχουν από το μέσο τους όρο. Έτσι, η διασπορά των υπολειπόμενων τιμών του macroblock μας λέει κατά πόσο τα pixels εμφανίζουν ομοιογένεια, ούτως ώστε να μπορούμε να αποφασίσουμε αν μιλάμε για ένα ή περισσότερα αντικείμενα, κάτι που θα δούμε αναλυτικά στη συνέχεια.

Οι αλγόριθμοι που έχουν αναπτυχθεί ως σήμερα (παραδείγματα υλοποιήσεων εδώ και εδώ) αν και επιτυγχάνουν πολύ καλά την αναγνώριση αντικειμένων, έχουν το αρνητικό πως επί το πλείστον είναι offline, δηλαδή εκτελούνται σε μη-πραγματικό χρόνο, δεχόμενοι ένα συνολικό αρχείο βίντεο σαν είσοδο και δίνουν σαν έξοδο το αποτέλεσμα ενώ επιπλέον οι περισσότεροι δουλεύουν σε εικόνα και όχι σε βίντεο. Ένας αλγόριθμος όμως που θα έτρεχε (για βίντεο) σε πραγματικό χρόνο θα άνοιγε νέες προοπτικές, αν σκεφτούμε πόσες κάμερες υπάρχουν γύρω μας καθημερινά και τις εφαρμογές που θα μπορούσαν να αναπτυχθούν από την πληροφορία των κινούμενων αντικειμένων σε αυτές.

Σκοπός της εργασίας

Ο σκοπός της εργασίας είναι η ανάπτυξη ενός αλγόριθμου ο οποίος θα αναγνωρίζει κινούμενα αντικείμενα σε ένα βίντεο και θα τρέχει σε πραγματικό χρόνο.

Μέσα από τα δύο παραπάνω μεγέθη, δηλαδή τα διανύσματα κίνησης και τη διασπορά των υπολειπόμενων pixels, μπορούμε να πάρουμε πολύτιμες πληροφορίες για το βίντεο που επεξεργαζόμαστε, ακόμη και κατά τη διάρκεια της συμπίεσης του, η οποία πλέον είναι δεδομένη σε κάθε βίντεο. Έτσι, όπως θα δούμε και στη συνέχεια, με τα δύο παραπάνω μεγέθη θα μπορέσουμε να αναπτύξουμε ένα αλγόριθμο ο οποίος θα αναγνωρίζει τα κινούμενα αντικείμενα και ο οποίος θα έχει σημαντικότερο πλεονέκτημά του την ταχύτητα.

Πέραν της αρχικής και λειτουργικής υλοποίησης του αλγόριθμου, αναγνωρίζουμε και την ανάγκη να τρέξει αυτός σε πραγματικό χρόνο. Αυτό αποτελεί και τη μεγαλύτερη πρόκληση στο εγχείρημά μας, μιας και ένας τέτοιος αλγόριθμος θα έχει πολλαπλές χρήσεις, όπως για παράδειγμα για κάμερες ασφαλείας οι οποίες θα αναγνωρίζουν τη μορφή ενός ανθρώπου (πιθανόν ληστή) σε ένα μαγαζί, ή κάμερες ελέγχου κυκλοφοριακής κίνησης (το «έξυπνο φανάρι») κ.α. Τέλος, μια απλή εφαρμογή σε ένα κανονικό κινητό με κάμερα, τύπου «smartphone» θα μπορεί να κοιτάει αν τα μήλα που περνάνε από τη γραμμή παραγωγής ενός εργοστασίου είναι αρκετά για να θεωρηθεί αυτή "υγιή" και αν περνάνε με τη σωστή ταχύτητα. Με τον καθημερινό πολλαπλασιασμό των κινητών με δυνατούς επεξεργαστές και κάμερα, ένας τέτοιος αλγόριθμος που θα τρέχει σε πραγματικό χρόνο φαίνεται πως μπορεί να έχει ποικίλες εφαρμογές, σημαντικού ενδιαφέροντος.

Για να επιτύχουμε αυτή τη αύξηση στη ταχύτητα εκτέλεσης θα χρησιμοποιήσουμε εξειδικευμένους αλγόριθμους ανεύρεσης κίνησης (motion estimation) που έχουν αναπτυχθεί τα τελευταία χρόνια, όπως οι Diamond search (αναφορά εδώ), Mvfast (αναφορά εδώ) και Pmvfast (αναφορά εδώ) αντί του κλασσικού και απαιτητικού σε χρόνο, αλγόριθμο Πλήρους Διερεύνησης (full search).

Επιπλέον θα χρησιμοποιήσουμε και εξειδικευμένες αρχιτεκτονικές, συγκεκριμένα τον επεκτάσιμο επεξεργαστή Xtensa της εταιρίας Tensilica, οι οποίες μπορούν να τροποποιηθούν έτσι ώστε να τρέχουν το κώδικά μας ακόμη πιο γρήγορα.

Διάρθρωση της Διπλωματικής Εργασίας

Στο 2ο κεφάλαιο αναλύεται η *φύση του βίντεο* και η σημαντικότερες συνιστώσες του όπως η κίνηση και η εκτίμησή της , τα διανύσματα κίνησης και η πληροφορία που μπορούν να δώσουν αυτά, απαραίτητο υπόβαθρο για να γίνει κατανοητός ο αλγόριθμος που θα παρουσιαστεί.

Στο 3ο κεφάλαιο θα παρουσιαστούν *διάφοροι αλγόριθμοι για motion estimation,* το οποίο είναι και το κομμάτι που καταναλώνει τον περισσότερο χρόνο στον αλγόριθμό μας, άρα και το κλειδί στην επίτευξη εκτέλεσης σε πραγματικό χρόνο.

Το 4ο κεφάλαιο εξηγεί πώς θα γίνει η *αναγνώριση των αντικειμένων*, τι πληροφορία μπορούμε να πάρουμε από τη διασπορά και τί από τα διανύσματα κίνησης καθώς και πώς θα γίνει ο *επιμερισμός (splitting)* των macroblocks και τι τεχνικές θα ακολουθήσουμε γι αυτό.

Το 5ο κεφάλαιο ασχολείται με το θέμα *συνένωσης blocks (merging*) και τις τεχνικές που επιτυγχάνεται με αυτό.

Το 6ο κεφάλαιο ασχολείται με τον τρόπο που έγινε η αναγνώριση αντικειμένων και των *χρωματισμό* αυτών, καθώς και άλλες τεχνικές που χρησιμοποιήθηκαν όπως αυτή της διαστολής (dilation).

Το 7ο κεφάλαιο διαπραγματεύεται τα *προβλήματα* που συναντά ο κώδικας αλλά και τους τρόπους που επιδιώκει να τα προσπεράσει.

Το 8ο κεφάλαιο παρουσιάζει την *αρχιτεκτονική που θα χρησιμοποιηθεί (Tensilica Xtensa)* για να τρέξουμε τον αλγόριθμο και τις δυνατότητες που έχει αυτή, καθώς και ο λόγος επιλογής της.

Στο 9ο αναλύονται οι *βελτιστοποιήσεις* που έγιναν στην αρχιτεκτονική Tensilica Xtensa για να τρέξει ο κώδικας γρηγορότερα.

Και στο 10ο και τελευταίο κεφάλαιο, θα *εξηγηθούν και θα αξιολογηθούν τα αποτελέσματα*, θα γίνει μια αποτίμηση για τον αλγόριθμο και μια αναφορά σε πιθανές μελλοντικές επεκτάσεις αυτού.

Κεφάλαιο 2ο:Βίντεο, Γενικοί Ορισμοί

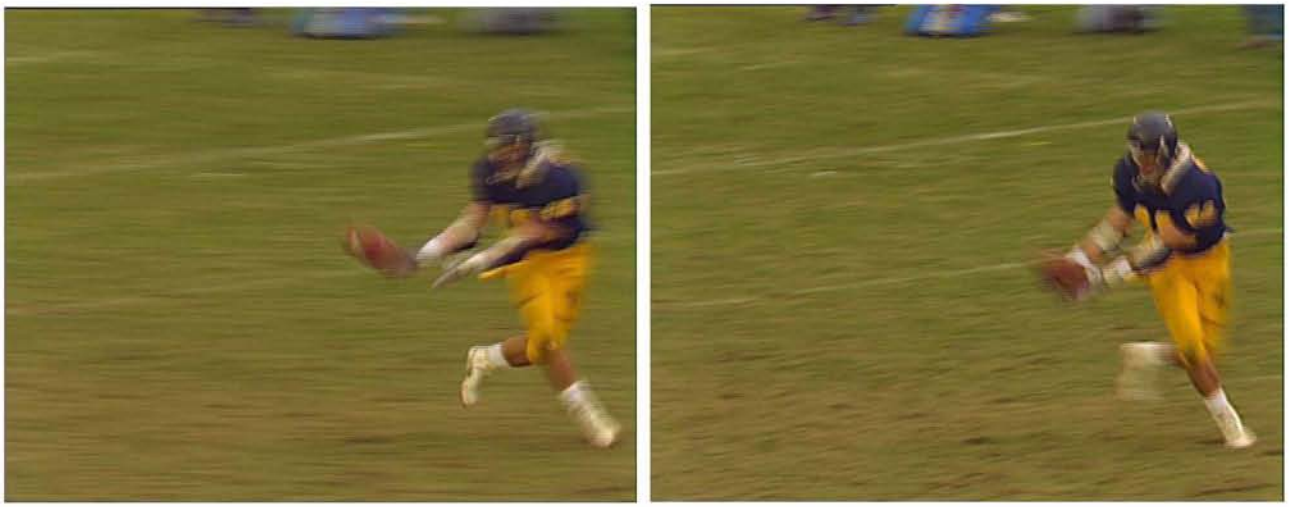
Κίνηση

Όπως μπορεί να αντιληφθεί και ο πιο άπειρος στην επεξεργασία βίντεο, το βίντεο δεν είναι τίποτα παραπάνω από εικόνες η μία μετά την άλλη, ή όπως τα λέμε, διαδοχικά καρέ (frames). Ένα τυπικό βίντεο (PAL ή NTSC) αποτελείται από 25 ή 30, αντίστοιχα, τέτοια frames το δευτερόλεπτο (fps-frames per second). Με αυτό το τρόπο το μάτι λόγω της "χαμηλής" απόκρισης που έχει στο χρόνο δεν μπορεί να αντιληφθεί διαδοχικές εικόνες και καταλαβαίνει μία συνεχή σκηνή η οποία είναι και αυτό που λέμε βίντεο.

Τα καρέ όμως δεν παύουν να υπάρχουν και έτσι σε ένα βίντεο, αν παγώσουμε την εικόνα και την προχωρήσουμε σε αργή κίνηση θα μπορέσουμε να τα αντιληφθούμε με μεγαλύτερη ακρίβεια. Το χαρακτηριστικό όμως που εισάγει το βίντεο και είναι δεδομένο σε όλα τα βίντεο (εκτός από τετριμμένες περιπτώσεις), είναι πως έχει μία συνέχεια, δηλαδή αν σε ένα βίντεο υπάρχει ένα αντικείμενο που κινείται, αναμένω αυτό να κινείται *σχετικά* γραμμικά, δηλαδή σε κάθε καρέ να έχει μετακινηθεί π.χ. 3 pixels , και λέμε σχετικά γιατί μπορεί κατά περίπτωση να αυξομειώνει την

ταχύτητα του. Για παράδειγμα αν βλέπω έναν αγώνα ποδοσφαίρου και σε ένα καρέ η μπάλα βρίσκεται στη θέση με συντεταγμένες (200,200) (οι συντεταγμένες αναφέρονται σε πόσα pixels απέχει από την άνω-αριστερά γωνία του βίντεο), αναμένω στο επόμενο καρέ να βρίσκεται κάπου εκεί γύρω, πχ σε απόσταση 0-50 pixel αναλόγως την ανάλυση της κάμερας με την οποία έχει γίνει η λήψη, αφού με κάμερα υψηλής ανάλυσης υπάρχουν πολλά περισσότερα pixels σε κάθε καρέ, και συνεπώς η αντίστοιχη κίνηση είναι μεγαλύτερη σε αριθμό pixels. Αυτό ισχύει γιατί μόνο έτσι το μάτι μπορεί να διακρίνει μια συνέχεια, αλλιώς αν η κίνηση του αντικειμένου σε διαδοχικά καρέ ήταν σε μια απόσταση, π.χ. 300 pixels, τότε ο παρατηρητής θα έβλεπε μια μπάλα να "τηλεμεταφέρεται" σπασμωδικά.

Και επειδή αυτό είναι κάτι που κανείς δεν θέλει, αλλά και κάτι που οι κλασσικές συσκευές λήψης βίντεο (βιντεοκάμερες) προσπαθούν να αποφύγουν, το στοιχείο της *συνέχειας της κίνησης* είναι κάτι που θα το θεωρήσουμε σαν δεδομένο για να το εκμεταλλευτούμε στην υλοποίησή μας, εξάλλου αυτό γίνεται κατά κανόνα και στους αλγόριθμους συμπίεσης βίντεο.



Σχήμα 1:Δύο διαδοχικά καρέ ενός βίντεο, όπου φαίνεται ότι το μεγαλύτερο ποσοστό της εικόνας παραμένει ίδιο.

Άρα για να επιτευχθεί η λογική του βίντεο τα καρέ πρέπει να έχουν μία συνέχεια, επομένως περιμένω (στη συντριπτική πλειοψηφία) κάθε καρέ να μοιάζει στο μεγαλύτερο ποσοστό του πχ 90% με το προηγούμενο καρέ. Τότε λοιπόν αυτό το υπόλοιπο 10% είναι και αυτό που λέμε *κίνηση* στο βίντεο. Δηλαδή στην ουσία η κίνηση είναι η *διαφορά* που έχει το *n* καρέ από το *n-1*.

Αυτή η κίνηση απομονώνεται με ειδικούς αλγορίθμους που έχουν αναπτυχθεί καθαρά για εφαρμογές συμπίεσης βίντεο και ονομάζονται γενικά *εκτίμηση κίνησης* (*motion estimation)* το οποίο θα καλύψουμε αναλυτικότερα παρακάτω στο κεφάλαιο.

Με αυτούς τους αλγόριθμους στους οποίους στηρίζεται η συμπίεση βίντεο, οι διάφοροι κωδικοποιητές αντί να αναθέτουν τα bits στο πώς είναι ένα καρέ(χρώματα), τα αναθέτουν στο τί διαφορά έχει αυτό από το προηγούμενο η οποία διαφορά είναι η κίνηση, και όπως είπαμε και πιο πάνω είναι στην ουσία ένα ποσοστό 10% σε σχέση με το 90% της υπόλοιπης εικόνας που δεν κινείται, έτσι μπορεί κάποιος να καταλάβει που βασίζεται η συμπίεση των βίντεο.

Αυτό που αποθηκεύεται δηλαδή, αντί να είναι το χρώμα των pixels του βίντεο είναι η κίνηση δηλαδή το *διάνυσμα της κίνησης (motion vector***)** και η διαφορά του από το προηγούμενο καρέ, η οποία όμως θα χρησιμοποιηθεί αργότερα στη διασπορά, άρα προς το παρόν τη ξεχνάμε.

Αυτοί λοιπόν οι motion vectors είναι ένα από τα δύο μεγέθη που θα μου δώσουν όλη την απαραίτητη πληροφορία για τον αλγόριθμό μου, οι οποίοι θα εξηγηθούν εκτενέστερα παρακάτω.

Εκτίμηση Κίνησης

Εκτίμηση κίνησης, γενικά στη συμπίεση βίντεο, ονομάζεται η διαδικασία κατά την οποία υπολογίζονται τα διανύσματα κίνησης και οι αντίστοιχες υπολειπόμενες διαφορές από το προηγούμενο καρέ. Τα διανύσματα κίνησης δίνουν από μόνα τους πληροφορία, ενώ η υπολειπόμενη διαφορά θα χρησιμοποιηθεί αργότερα για την εύρεση των διασπορών των macroblocks.

Η εκτίμηση κίνησης είναι το κύριο υπολογιστικό κομμάτι στην κωδικοποίηση βίντεο και το αντίστοιχο, που ονομάζεται απόδοση κίνησης (motion compensation) είναι ένα από τα κύρια κομμάτια στην αποκωδικοποίηση βίντεο.

Υπάρχουν πολλοί αλγόριθμοι εκτίμησης κίνησης, οι οποίοι θα αναλυθούν στο επόμενο κεφάλαιο, αλλά εδώ θα εξετάσουμε τον πρώτο και κλασσικό, αυτό της *εξαντλητικής/πλήρης διερεύνησης (full search).*

Έστω ότι έχουμε δύο διαδοχικά καρέ όπως αυτά φαίνονται παρακάτω:

Σχήμα 2: Το πρώτο (προγενέστερο χρονικά, στα αριστερά) καρέ λέγεται καρέ αναφοράς (reference picture) και το δεύτερο (μεταγενέστερο χρονικά, στα δεξιά) τρέχον καρέ (current picture), αυτό σημαίνει ότι θα υπολογίσουμε τις διαφορά του δεύτερου σε σχέση με το πρώτο.

Η λογική της εκτίμησης κίνησης είναι για κάθε macroblock (MB, κομμάτι 16x16 pixels) του τρέχοντος καρέ, στην γύρω περιοχή, με ποιό κομμάτι 16x16 μοιάζει περισσότερο στο καρέ αναφοράς. Για το κριτήριο της ομοιότητας των macroblocks θα χρησιμοποιήσουμε το SAD-Sum of absolute differences ή άθροισμα απόλυτων τιμών το οποίο και θα εξηγήσουμε παρακάτω. Ο πρώτος και πιο διαδεδομένος αλγόριθμος για εκτίμηση κίνησης είναι το full search.

Ο αλγόριθμος full search δουλεύει ως εξής:

Υπολογίζει το *SAD (sum of absolute differences)* του macroblock στη θέση (i,j) στο τρέχον καρέ και του macroblock στο καρέ αναφοράς στη θέση (i,j) καθώς επίσης και τα SAD του ίδιου αυτού macroblock στη θέση (i,j) στο τρέχον καρέ με το macroblock στη θέση (i+1,j) στο καρέ αναφοράς, μετά με το (i+2,j) macroblock στο καρέ αναφοράς και γενικά με όλα τα macroblock του καρέ αναφοράς σε μία δεδομένη περιοχή (συνήθως στις θέσεις +-15 pixels οριζοντίως και καθέτως σε σχέση με τη θέση του macroblock στο τρέχον καρέ) και κρατάει το macroblock, και την αντίστοιχη μετατόπιση του καρέ αναφοράς που έδωσε το μικρότερο SAD.

Το SAD ή *άθροισμα απόλυτων διαφορών* είναι η πράξη που χρησιμοποιείται περισσότερο στη συμπίεση βίντεο, καθώς και στον αλγόριθμό μας, άρα όπως θα δούμε και αργότερα η βελτιστοποίηση του είναι και το κλειδί στην βελτιστοποίηση όλου του αλγόριθμου.

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|$$

Το SAD ορίζεται ως:

όπου W είναι το πλαίσιο που θα κινηθεί η μετατόπιση των (i,j) και όπως είπαμε είναι συνήθως ένα

παράθυρο από (-15,-15) έως (+15,+15) (συνολικά, 31x31 = 961 υπολογισμοί SAD).

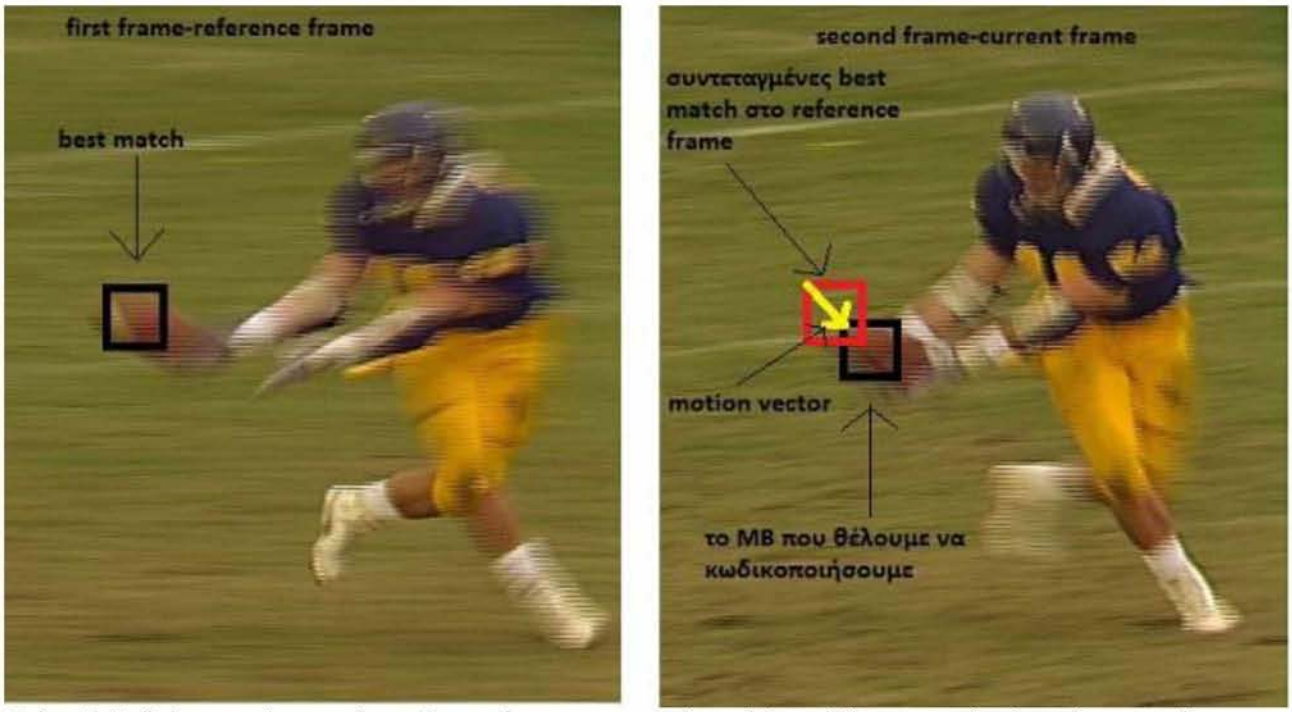
Σχήμα 3: Το 16x16 MB στο τρέχον καρέ που φαίνεται στο κόκκινο πλαίσιο δεξιά, όταν θα ψάξει να βρει το μικρότερο SAD στην γύρω περιοχή του καρέ αναφοράς που φαίνεται αριστερά, αυτό θα βρεθεί στο κόκκινο πλαίσιο του αριστερά καρέ, το οποίο είναι και αυτό που ονομάζεται βέλτιστο (best match). Με άλλα λόγια προσπαθούμε να βρούμε στο προηγούμενο χρονικά καρέ, το MB που μοιάζει πιο πολύ με το MB στο καρέ που ακολουθεί χρονικά.

Το κύριο *μειονέκτημα* του full search όπως θα δούμε και παρακάτω όμως είναι πως υπολογίζει τα SAD για όλα τα (i,j) στο πλαίσιο και άρα, αν και υπολογίζει όντως το πραγματικό βέλτιστο στη περιοχή, είναι αργός σαν αλγόριθμος.

Διανύσματα Κίνησης

Εφόσον έχει εκτελεστεί ο αλγόριθμος εύρεσης κίνησης, όπως ο full search ή οποιοσδήποτε άλλος αλγόριθμος που θα δούμε παρακάτω, το διάνυσμα κίνησης δεν είναι τίποτα παραπάνω από την *διαφορά της θέσης του αρχικού macroblock στο τρέχον καρέ από τη θέση του best match που βρήκαμε στο καρέ αναφοράς*. Με άλλα λόγια το διάνυσμα κίνησης είναι όντως ένα διάνυσμα δύο διαστάσεων (mvx, mvy) που μας δείχνει πώς κινήθηκε ή καλύτερα πού πήγε το macroblock στο διάστημα του χρόνου που πέρασε μεταξύ της λήψης των δύο καρέ.

Σχήμα 4: Μία κλασσική αντίληψη του διανύσματος κίνησης, το φανταζόμαστε σαν ένα βελάκι που δείχνει προς το σημείο που υπάρχει το MB στο τρέχον καρέ και ξεκινά από το MB(αντίστοιχες συντεταγμένες) που βρέθηκε η βέλτιστη επιλογή στο καρέ αναφοράς.



Σχήμα 5: Το διάνυσμα κίνησης είναι η διαφορά των συντεταγμένων (i,j) της βέλτιστης επιλογής από το αρχικό macroblock, όπως φαίνεται στην εικόνα. Το πρόσημο είναι αποτέλεσμα σύμβασης.

Εδώ ίσως πλέον να έχει δημιουργηθεί η εξής αμφιβολία. Κάποιος θα μπορούσε να πει πως ίσως το αντικείμενο που υπάρχει στο macroblock του τρέχοντος καρέ να εμφανίστηκε για πρώτη φορά ή να άλλαξε μορφή ή χρώμα. Σε μια τέτοια περίπτωση, ο αλγόριθμος είναι λογικό να παραπλανηθεί και να βρει σαν βέλτιστη επιλογή στο καρέ αναφοράς ένα MB το οποίο θα έχει μεγάλη υπολειπόμενη διαφορά από το MB στο τρέχον καρέ. Κάτι τέτοιο δε σημαίνει ότι ο αλγόριθμος αποτυγχάνει, απλά πλέον δεν ισχύει η συνθήκη μας ότι το βίντεο έχει μια συνέχεια στο κομμάτι του συγκεκριμένου καρέ και θα χρειαστούν περισσότερα bits για να κωδικοποιηθεί το βίντεο αν μιλάμε για συμπίεση, ή το ότι δεν θα πάρουμε την απαραίτητη πληροφορία για να αναγνωρίσουμε το αντικείμενο όταν μιλάμε για την εφαρμογή της εργασίας μας. Άρα η εκτίμηση κίνησης όπως θα δούμε και παρακάτω έχει κάποιες εγγενείς αδυναμίες και οι συνέπειες του θα αναλυθούν αργότερα.

Πληροφορία των Διανυσμάτων Κίνησης

Τώρα που κατανοήσαμε τι είναι το διάνυσμα και πώς κατασκευάζεται, είναι εύκολο να κατανοήσουμε και την συμβολή του. Αρχικά μιλάμε για διανύσματα κίνησης (mv) από macroblocks (MB) 16x16, έτσι αν ένα MB έχει, για παράδειγμα, mv (0,0) και το διπλανό του έχει mv (3,3), άμεσα βγαίνει το συμπέρασμα ότι το πρώτο αναφέρεται σε ένα αντικείμενο που είναι ακίνητο και το δεύτερο σε ένα που κινείται κάτω και δεξιά, έτσι κοιτώντας όλα τα γύρω MB, καταλαβαίνουμε πως όσα MB έχουν mv (0,0) είναι, κατά πάσα πιθανότητα ακίνητα και γι'αυτό ανήκουν στην κατηγορία του φόντου (background) και όσα έχουν (3,3) ανήκουν σε κάποιο αντικείμενο που κινείται, ας το ονομάσουμε «αντικείμενο #1» το οποίο κινείται προς τα κάτω και δεξιά. Τώρα, αν στην πορεία ανακαλύψουμε ένα διπλανό MB με mv (-1,0), πρόκειται προφανώς για ένα νέο αντικείμενο («αντικείμενο #2»), το οποίο κινείται προς τα αριστερά.

Έτσι αρχίζει να φαίνεται η υψηλή πληροφορία που μπορεί να δώσει το διάνυσμα κίνησης ακόμη και σε MB 16x16, το οποίο προσφέρει ένα σχετικά χαμηλό επίπεδο λεπτομέρειας.

gebardas@uth.gr

Κεφάλαιο 3ο:Αλγόριθμοι Εκτίμησης Κίνησης

Πλήρης Διερεύνηση (Full Search)

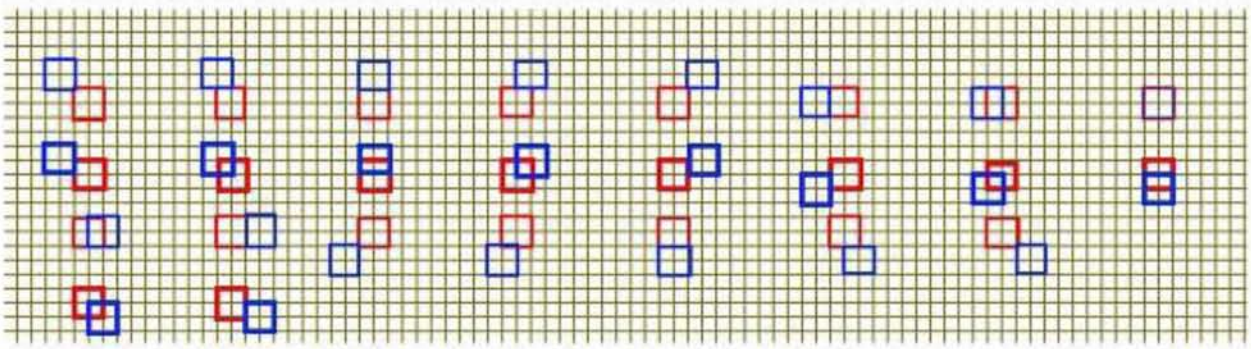
Η Πλήρης Διερεύνηση (full search) είναι ο πρώτος και πιο διαδεδομένος αλγόριθμος για εκτίμηση κίνησης μιας και χρησιμοποιήθηκε στα mpeg-1 και mpeg-2.

Γενικά όπως είπαμε και παραπάνω, βασίζεται στην λογική του να εξετάζονται *όλες* οι θέσεις στο τρέχον καρέ σε ένα πλαίσιο, x+-εμβέλεια,y+-εμβέλεια, με εμβέλεια συνήθως +/-15, άρα είναι εμφανές οτι θα χρειαστεί να υπολογίσουμε ένα μεγάλο αριθμό από SAD, για την ακρίβεια 31x31=*961* και από αυτά να επιλέξουμε το ελάχιστο για βέλτιστη επιλογή (best match).

Για να κατανοήσουμε καλύτερα τον αλγόριθμο αρκεί να πούμε πως άν το macroblock(MB) στο τρέχον καρέ είναι στο (255,255), ο αλγόριθμος θα εξετάσει ένα ένα τα macroblocks 16x16 που το πάνω αριστερά pixel τους είναι το :

1)(239,239), 2)(240,239), 3)(241,239) κ.ο.κ μέχρι να καταλήξει στο τελικό 961)(271,271) του καρέ αναφοράς.

 gebardas@uth.gr



Σχήμα 6: Παράδειγμα MB 2x2 και με κόκκινο χρώμα το MB του τρέχοντος καρέ, τότε το MB του καρέ αναφοράς θα είναι το μπλε.

Έτσι όπως θα έχει γίνει ήδη φανερό το κύριο μειονέκτημα του FS είναι πως εξετάζοντας όλες τις θέσεις καταναλώνει πολύ χρόνο, και άρα είναι πολύ αργό, έτσι αποκλείεται για τον αλγόριθμο μάς.

Παράλληλα όμως, το γεγονός πως εξετάζει όλες τις θέσεις μας δίνει τη σιγουριά πως θα βρει εγγυημένα το best match τουλάχιστον στο πλαίσιο που ψάχνει, γιατί φυσικά κανένας δεν μας εγγυάται ότι το πραγματικό best match είναι σε απόσταση -15/+15 ή όσο είναι η εμβέλεια του αλγόριθμου.

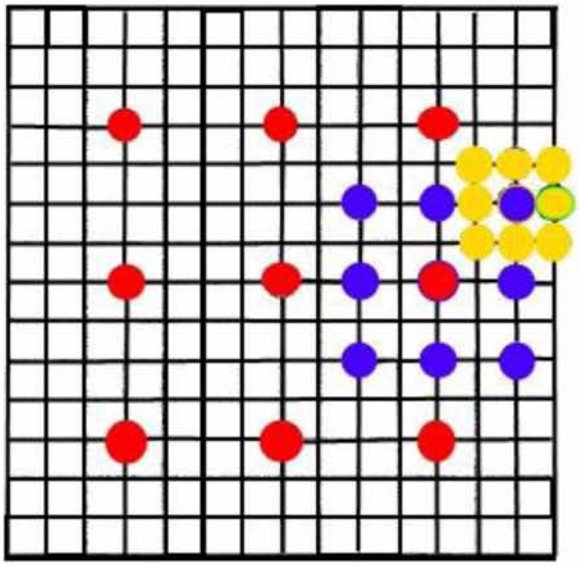
Διερεύνηση Τριών ή N Βημάτων(Three or N Step Search)

Διερεύνηση Τριών Βημάτων (Three Step Search) ή αλλιώς Διερεύνηση N Βημάτων (N-Step Search) (για γενίκευση *N* βημάτων), έχει πάρει το όνομά της από την ιδιότητά της να εκτελείται σε βήματα, όπου σε κάθε βήμα μικραίνει το πλαίσιο αναζήτησης.

Η κύρια διαφορά του TSS από τον FS είναι ότι δεν εξετάζονται όλες οι θέσεις του πλαισίου, παρά μόνο οι οχτώ ακραίες και η κεντρική, τότε βρίσκεται η ελάχιστη από αυτές και η διάσταση του πλαισίου υποδιπλασιάζεται, συνεχίζοντας την αναζήτηση από εκεί που βρέθηκε η καλύτερη επιλογή.

Αναλυτικότερα, ο αλγόριθμος ξεκινά από την θέση (0,0) του καρέ αναφοράς και ψάχνει τις ακραίες οχτώ θέσεις γύρω του που απέχουν +/-S(εμβέλεια) και την κεντρική, με S= 2^(N-1). Από αυτές διαλέγει αυτή με το ελάχιστο SAD και αφού κάνει S=S/2 συνεχίζει αναδρομικά για την θέση

που βρήκε το ελάχιστο, μέχρι S=1.



Σχήμα 7: Με κόκκινο παρατηρώ τα sad του 1ου βήματος, με μπλε αυτά του 2ου και με κίτρινο αυτά του 3ου βήματος. Εδώ φυσικά ισχύει N=3, άρα όντως μιλάμε για Three Step Search.

Για να συγκρίνουμε με τον FS αρκεί να παρατηρήσουμε πως για να έχουμε την ίδια διάσταση πλαισίου αναζήτησης αρκεί να θέσουμε N=5(αριθμός βημάτων) και έτσι θα έχουμε S=16. Σε αυτή τη περίπτωση θα έχουμε πέντε βήματα και άρα 5x9=45 εκτελέσεις SAD, άρα αν τη συγκρίνουμε με τις 961 της FS, βλέπουμε πως η N Step Search είναι 961/45=21 φορές πιο γρήγορη.

Σίγουρα σε θέμα αξιοπιστίας αποτελέσματος χάνουμε σε σχέση με την FS, αλλά όχι πολύ, μιας και είναι λογικό πως όταν βρίσκουμε ένα ελάχιστο σε μία θέση του 1ου βήματος, στην πλειοψηφία των περιπτώσεων η πραγματική βέλτιστη επιλογή θα είναι κάπου εκεί γύρω, άρα ρίχνουμε το S για να ψάξουμε με μεγαλύτερη ακρίβεια.

Μία παραλλαγή του N Step Search είναι το *Cross Search*, το οποίο διαφέρει μόνο στο γεγονός πως αντί για εννιά σημεία, εξετάζονται τα πέντε ακραία έτσι ώστε να σχηματίζεται ένα X.

Λογαριθμική Αναζήτηση (Logarithmic Search)

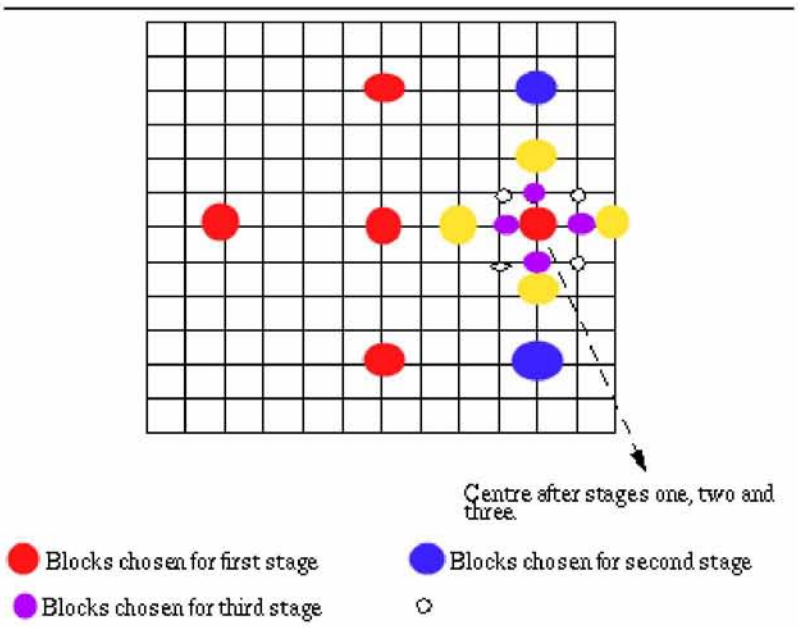
Ο αλγόριθμος κάθε φορά εξετάζει πέντε σημεία, το κεντρικό και τα τέσσερα ακραία που σχηματίζουν σταυρό. Από τα πέντε αυτά σημείο διαλέγουμε αυτό που θα δώσει το ελάχιστο SAD για να το χρησιμοποιήσουμε στην επόμενη αναδρομή και στην αναδρομή που θα βρούμε σαν ελάχιστο το κεντρικό, κάνω S=S/2(εμβέλεια) και συνεχίζουμε.

Το S είναι το βήμα (step size όπως) είδαμε παραπάνω και αρχικοποιείται, αναλόγως με την ακρίβεια που θέλουμε στο αποτέλεσμα. Αντιλαμβανόμαστε έτσι ότι για μεγαλύτερο step size έχουμε μεγαλύτερη αξιοπιστία για το best match που θα βρω αλλά και περισσότερα SAD άρα περισσότερες πράξεις.

Ο αλγόριθμος θα είναι σίγουρα πιο γρήγορος από τον FS αλλά θα έχει μικρότερη αξιοπιστία, όπως και ο NSS..

Η κύρια διαφορά με τον NSS είναι πως εδώ υποδιπλασιάζουμε το βήμα μόνο όταν βρεθεί η βέλτιστη επιλογή στο κέντρο του σχήματος αναζήτησης.

.



Σχήμα 8: Παράδειγμα για αρχικό S=4(εμβέλεια/range), με τα κόκκινα 1ο βήμα τα μπλε 2ο,τα κίτρινα 3ο και τα μοβ 4ο. Στο 3ο βήμα S=2 και στο 4ο S=1.

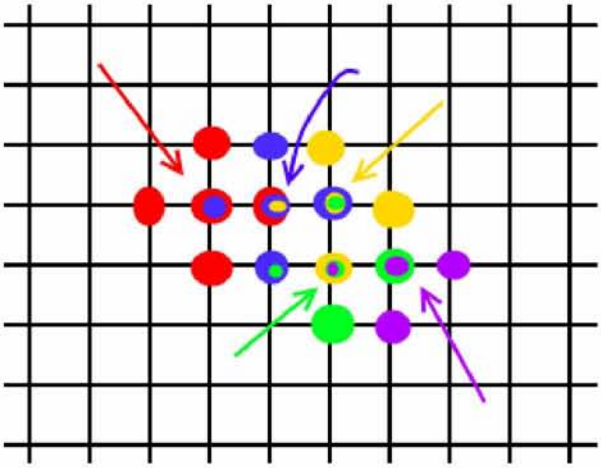
Αναζήτηση Διαμαντιού (Diamond Search)

Η Αναζήτηση Διαμαντιού διαφέρει με όλους τους παραπάνω αλγόριθμους εκτίμησης κίνησης γιατί είναι ο πρώτος που υιοθετεί τη λογική της *απότομης καθόδου* (gradient descent).

Η *απότομη κάθοδος* είναι ένας κλασσικός αλγόριθμος ο οποίος προσεγγίζει τοπικά ελάχιστα ή μέγιστα συναρτήσεων σε βήματα, τείνοντας όλο και περισσότερο προς τη λύση. Βασίζεται στη λογική ότι, όσο τείνουμε προς το ελάχιστο/μέγιστο η συνάρτηση συνέχεια θα μειώνεται/αυξάνεται αντίστοιχα. Αντίστοιχα και ο DS μας λέει πως περιμένουμε ότι όσο τείνουμε προς το best match θα

βρίσκουμε όλο και μικρότερα SAD και το αντίστροφο. Για παράδειγμα, ξεκινάμε από τη θέση (0,0) και εξετάζουμε τα SAD των τεσσάρων γειτονικών MB και του κεντρικού, αν το μικρότερο SAD είναι αυτό του κεντρικού ο αλγόριθμος τερματίζει. Διαφορετικά καλείται αναδρομικά με κεντρικό τώρα αυτό που έχει το μικρότερο SAD.

Ο DS είναι ο πρόγονος των MVFAST και PMVFAST που θα αναφέρουμε παρακάτω.



Σχήμα 9: Παράδειγμα DS, το κόκκινο αντιπροσωπεύει το πρώτο βήμα,το μπλε το 2ο, το κίτρινο το 3ο, το πράσινο το 4ο και το μοβ το 5ο. Τα βελάκια των αντίστοιχων χρωμάτων δείχνουν το εκάστοτε ελάχιστο.

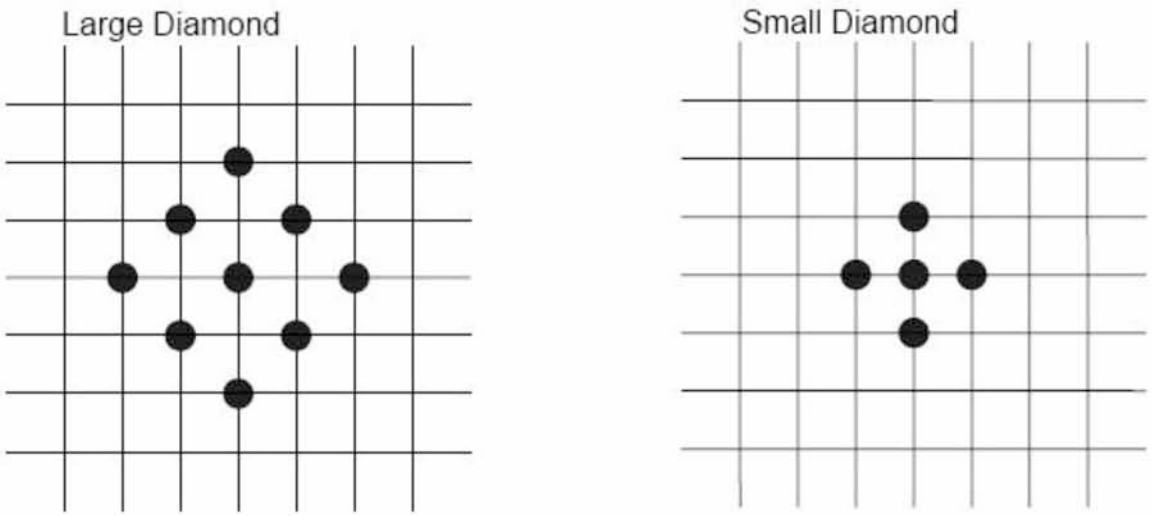
Ο DS είναι ένας πολύ αποδοτικός αλγόριθμος που δίνει τη βέλτιστη επιλογή με μεγάλη αξιοπιστία, μιας και η λογική της απότομης καθόδου έχει αποδειχθεί ότι δουλεύει στο μεγαλύτερο ποσοστό των περιπτώσεων. Έχει όμως το αρνητικό πως κάνει πολλές επαναλήψεις μέχρι να καταλήξει στο τελικό best match ειδικά σε περιπτώσεις όπου το best match θα απέχει πολύ από το macroblock που επεξεργαζόμαστε. Επιπλέον σε πολλές περιπτώσεις παρατηρείται πως ο αλγόριθμος μπορεί να έχει φτάσει σε ένα πολύ μικρό SAD (π.χ. 95 ή και χαμηλότερο) και να συνεχίζει να εκτελείται αναδρομικά απλά επειδή βρίσκει όλο και λίγο μικρότερα SAD, τα οποία όμως έχουν μικρή διαφορά με το προηγούμενο και εν τέλει δεν προσφέρουν πολλά, απλά σπαταλάνε χρόνο στην υλοποίηση.

Τεχνική Αναζήτησης Προσαρμοσμένου Πεδίου Διανύσματος Κίνησης (Motion Vector Field Adaptive Search Technique-MVFAST)

Ο MVFAST αλγόριθμος βασίζεται στην Αναζήτηση Διαμαντιού(DS) και αποτελεί μια σημαντική βελτίωση αυτής.

Είναι ένας αρκετά πολύπλοκος αλγόριθμος ο οποίος βασίζεται όπως και ο DS στη λογική της απότομης καθόδου αλλά χρησιμοποιεί κάποια κριτήρια και κατώφλια για να αποφύγει την μεγάλη πολυπλοκότητα του DS και να συγκρατήσει την υψηλή αξιοπιστία του.

Αναλυτικά ο αλγόριθμος χρησιμοποιεί δύο σχήματα αναζήτησης (search patterns), ένα μικρό σχήμα διαμαντιού(small diamond), ίδιο με αυτό του αλγόριθμου DS και ένα μεγάλο σχήμα διαμαντιού (large diamond), μεγαλύτερο σε έκταση και τρεις καταστάσεις για να χαρακτηρίσει την κινητικότητα (motion activity) του καρέ. Έτσι χρησιμοποιεί αναλόγως με την κατάσταση κίνησης και το ανάλογο σχήμα αναζήτησης (pattern).



Σχήμα 10: Η διαφορά του large diamond από το small

.

Πρώτον, επειδή παρατηρείται πως αρκετά macroblocks (MB) είναι ακίνητα σε πολλά καρέ, φροντίζει να τα αναγνωρίσει σύντομα χρησιμοποιώντας ένα κατώφλι τα (στην υλοποίησή μας αυτό είναι ανάλογο με το μέγεθος του MB, δηλαδή για μικρότερο MB έχει μικρότερη τιμή) και άν το (0,0) έχει SAD μικρότερο από αυτό τότε θεωρεί σαν βέλτιστη επιλογή (best match) το (0,0). Με

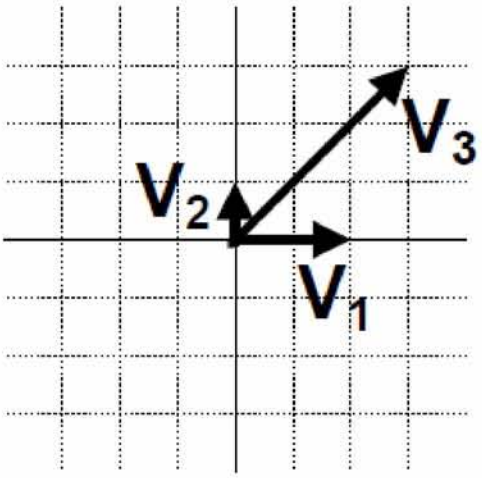
αυτό τον τρόπο αποφεύγονται οι διαδοχικές κλήσεις για μικρά SAD.

Χαρακτηρίζει την κινητικότητα σε μικρή (low),μέση (medium),μεγάλη (large) κοιτάζοντας τα διανύσματα κίνησης των γειτονικών MB, συγκεκριμένα του αριστερού, του πάνω και του πάνω-δεξιά τα οποία έχουν βρεθεί προηγουμένως. Έτσι η κινητικότητα χωρίζεται:

- Low αν L<=L1 (όπου L,L1,L2 ορίζονται παρακάτω)
- Medium αν L1<L<=L2
- High αν L>L2

Όπου L=max(x+y) για όλα τα motion vectors των MB που είπαμε παραπάνω και L1,L2 κατώφλια που ορίζουμε εμείς, συνήθως 1 και 2 αντίστοιχα. Αυτή η διάκριση σε καταστάσεις κινητικότητας γίνεται γιατί για μικρή κίνηση αναμένω η βέλτιστη επιλογή να είναι κάπου κοντά, ενώ για μεγάλη αναμένω να είναι μακριά. Μιας και ισχύει η λογική πως σε πολλές περιπτώσεις αναμένουμε ένα MB να έχει παρόμοιο mv με αυτό των γειτόνων του.

Η διαφορά των τριών καταστάσεων είναι το πιο σχήμα αναζήτησης χρησιμοποιούνε, μιάς και η μεσαία χρησιμοποιεί το μεγάλο, ενώ οι άλλες δύο το μικρό. Επίσης όμως, η μικρή ξεχωρίζει από την μεγάλη, μιας και η αναζήτηση της μικρής ξεκινά από το (0,0), ενώ της μεγάλης από το MB του συνόλου V που δίνει το μικρότερο SAD.



Σχήμα 11: Στο παράδειγμα μας έχω v1=(2,0),v2=(1,0),v3=(3,3),άρα lv1=2, lv2=1, lv3=6 άρα και L=MAX{lv1, lv2, lv3}= 6 όπου V1,V2,V3 τα motion vectors του πάνω-δεξιά,πάνω και αριστερά macroblocks.

Το κέντρο ψαξίματος είναι το (0,0) αν έχουμε χαμηλή ή μέση κινητικότητα, αν όμως έχουμε μεγάλη, τότε χρησιμοποιεί σαν κέντρο το διάνυσμα κίνησης του MB που δίνει το μικρότερο SAD, από τα παραπάνω MB που αναφέραμε. Αυτό δείχνει πως ο αλγόριθμος προσπαθεί να προβλέψει το διάνυσμα κίνησης του MB κοιτώντας τα γειτονικά, μιας και για μεγάλη κινητικότητα αναμένω να κάνω πολλές επαναλήψεις μέχρι να φτάσουμε στη βέλτιστη επιλογή.

Ο αλγόριθμος διαλέγει μικρό σχήμα διαμαντιού αν η κινητικότητα είναι χαμηλή ή υψηλή και μεγάλο σχήμα διαμαντιού αν είναι μέση, ως εξής:

- **Μικρό Σχήμα Διαμαντιού (Small diamond)**: Ξεκινούμε από το κέντρο ψαξίματος και αν το ελάχιστο SAD βρεθεί στο κέντρο τότε αυτό είναι η βέλτιστη επιλογή, αλλιώς καλείται αναδρομικά για αυτό που βρέθηκε ελάχιστο, ότι κάνει και ο DS δηλαδή.
- **Μεγάλο Σχήμα Διαμαντιού (Large diamond)**: Ξεκινούμε από το κέντρο ψαξίματος και ελέγχουμε τα MB που είδαμε στην εικόνα παραπάνω, αν το ελάχιστο είναι το κεντρικό τότε καλούμε και ένα small diamond στο κεντρικό, αλλιώς καλούμε αναδρομικά πάλι large diamond με κεντρικό αυτό που έδωσε το ελάχιστο SAD.

<u>Πλεονεκτήματα του MVFAST:</u>

1. **Απλότητα**: Υλοποιείται με λίγες γραμμές κώδικα παραπάνω από την Αναζήτηση Διαμαντιού (DS).
2. **Σταθερότητα**: Δεδομένη, μιας και βασίζεται στον ήδη πολύ σταθερό DS, αλλά και

αποτέλεσμα μακροχρόνιας χρήσης σε υλοποιήσεις.

3. **Απόδοση Κόστους**: Λέμε ότι είναι αποδοτικός στο κόστος γιατί έχει σχεδιαστεί με ένα έξυπνο τρόπο ο οποίος του επιτρέπει να μην περνάει από το ίδιο σημείο αναζήτησης δύο φορές, έτσι δεν χρειάζεται να αποθηκεύει ποια σημεία έχει επισκεφθεί.
4. **Κλιμάκωση**: Είδαμε πως ο MVFAST έχει τρεις καταστάσεις για τον χαρακτηρισμό της κινητικότητας. Αναλόγως την εφαρμογή μπορούμε να χρησιμοποιήσουμε σαν δεδομένα κάποια από τις τρεις, χρησιμοποιώντας συγκεκριμένες τιμές στα κατώφλια L1,L2(π.χ. για L1,L2=0 θα εκτελεί πάντα λειτουργία υψηλής κινητικότητας).
5. **Υψηλή Αποδοτικότητα**: Σε σύγκριση με άλλους γρήγορους αλγόριθμους εκτίμησης κίνησης, ο MVFAST είναι ο γρηγορότερος. Παρά την υψηλή του ταχύτητα, επιτυγχάνει σχεδόν ίδια ποιότητα βίντεο με την Πλήρη Διερεύνηση (Full search). Παράλληλα παρατηρήθηκε πως ο FS όταν βρει ένα μεγάλο διάνυσμα κίνησης π.χ (-20,19), ο MVFAST θα δώσει χαμηλότερο, στην αντίστοιχη περίπτωση, π.χ (-15,15) το οποίο συνήθως έχει παρόμοιο sad με το 1ο, και έτσι διευκολύνει και στην συμπίεση βίντεο, κόβοντας αυτά τα μεγάλα διανύσματα κίνησης που απαιτούνε περισσότερα bits για κωδικοποίηση.
6. **Αξιοπιστία**: Ο MVFAST παρουσιάστηκε πρώτη φορά το Δεκέμβριο του 1999 και η υψηλή του απόδοση αποδείχθηκε με τη βοήθεια 30 πειραμάτων από το Video Group. Επιπλέον η ISG έκανε ακόμη 24 πειράματα τα οποία επιβεβαίωσαν την υψηλή του ταχύτητα αλλά και ποιότητα.

Παραδείγματα τρεξίματος των δύο αλγορίθμων σε διάφορα βίντεο:

Sequence	Bitrate	Frame Rate	Search	PSNR-Y		PSNR-U		PSNR-V		SEARCH POINTS	
	(kbps)	(fps)	Range	FS	MVFAST	FS	MVFAST	FS	MVFAST	FS	MVFAST
container	10	7.5	16	29.81	29.78	37.54	37.50	36.60	36.64	7501824	33149
container	10	7.5	32	29.72	29.79	37.55	37.49	36.57	36.59	27142090	33486
Hall monitor	10	7.5	16	30.35	30.32	36.38	36.28	39.57	39.51	7501824	31418
Hall monitor	10	7.5	32	30.29	30.23	36.24	36.46	39.49	39.63	27142090	31370
Mother Dtr	24	10	16	34.80	34.76	40.23	40.24	41.02	40.98	10036224	48611
Mother Dtr	24	10	32	34.81	34.68	40.28	40.27	40.98	41.02	36311715	49604
Silence	24	10	16	30.85	30.88	35.56	35.68	36.93	36.95	10036224	71268
Silence	24	10	32	30.78	30.88	35.42	35.69	36.90	37.05	36311715	71506
Coastguard	48	10	16	28.88	28.83	40.14	40.20	42.07	42.15	10036224	99468
Coastguard	48	10	32	28.90	28.82	40.02	40.20	41.88	42.10	36311715	99977
News	48	7.5	16	31.84	31.80	35.72	36.06	37.31	37.49	30007296	168005
News	48	7.5	32	31.90	31.75	35.79	35.93	37.35	37.46	114227658	171416
News	112	15	16	34.05	33.97	38.04	37.93	38.93	38.91	60420096	264061
News	112	15	32	34.03	33.96	37.93	37.95	38.85	38.85	229998933	266329
Foreman	112	10	16	30.04	29.91	36.78	36.93	37.53	37.70	40144896	474898
Foreman	112	10	32	30.37	30.18	36.88	36.96	37.56	37.81	152818083	497026
Coastguard	112	10	16	27.03	27.05	38.87	39.12	41.65	41.64	40144896	413640
Coastguard	112	10	32	27.06	27.10	38.64	39.08	40.99	40.99	152818083	413316
Foreman	512	15	16	34.51	34.43	40.25	40.21	41.47	41.45	56770560	580020
Foreman	512	15	32	34.84	34.70	40.56	40.45	41.75	41.69	216106380	589759
Foreman	512	15	48	34.88	34.72	40.62	40.48	41.79	41.72	461637680	591276
Foreman	1024	30	16	35.47	35.37	41.05	40.96	42.36	42.29	113541120	902601
Foreman	1024	30	32	35.53	35.41	41.13	41.01	42.43	42.34	432212760	911217
Foreman	1024	30	48	35.51	35.40	41.11	41.00	42.40	42.32	923275360	911377
Tennis	1024	30	16	34.98	34.92	41.89	41.81	41.01	40.93	94617600	519726
Tennis	1024	30	32	35.00	34.92	41.91	41.81	41.02	40.93	358185240	522012
Tennis	1024	30	48	34.97	34.92	41.88	41.82	41.01	40.93	760543840	521379
Tennis	2048	30	16	37.95	37.90	43.47	43.41	42.98	42.93	94617600	480644
Tennis	2048	30	32	37.95	37.90	43.47	43.42	42.97	42.92	358185240	482591
Tennis	2048	30	48	37.94	37.89	43.45	43.40	42.96	42.91	760543840	483244

Σχ.13:Για κάθε βίντεο βλέπουμε το birtate του, το frame rate, το search range είναι η εμβέλεια που ψάχνει ο κάθε αλγόριθμος, τα PSNR(Peak Signal To Noise Ratio) σε Y,U,V, και τα σημεία αναζήτησης (search points) που είναι σε πόσα σημεία έψαξε, ή με άλλα λόγια πόσα sad έκανε. Παράρτημα από το άρθρο «Performance report of Motion Vector Field Adaptive Search Technique» από τους Kai-Kuang Ma και Prabhudev Irappa Hosur, εδώ.

Συμπεράσματα: Παρατηρούμε πρώτα από όλα, πως για εμβέλεια αναζήτησης=16, το PSNR των δύο μεθόδων είναι πρακτικά ίδιο, ενώ για εμβέλεια=32 είναι ελάχιστα καλύτερη η Πλήρης Διερεύνηση (Full Search), το οποίο είναι αναμενόμενο. Παράλληλα όμως παρατηρούμε πως η βελτίωση στη ταχύτητα του mvfast είναι τεράστια, μιας και για εμβέλεια=16 είναι πάνω από 220 φορές ταχύτερο από το full search, ενώ για εμβέλεια=32 γίνεται 810 φορές ταχύτερο, μία ταχύτητα η οποία είναι σημαντικότατη σε σχέση με την αμυδρή απώλεια σε ποιότητα(psnr).

Τεχνική Αναζήτησης Πρόβλεψης Προσαρμοσμένου Πεδίου Διανύσματος Κίνησης (PMVFAST-Predictive MVFAST)

Όπως φαίνεται και από το όνομά του, ο PMVFAST είναι μια βελτιωμένη έκδοση του MVFAST που βασίζεται στην πρόβλεψη (predictive). Οι κύριες διαφορές είναι πως στους

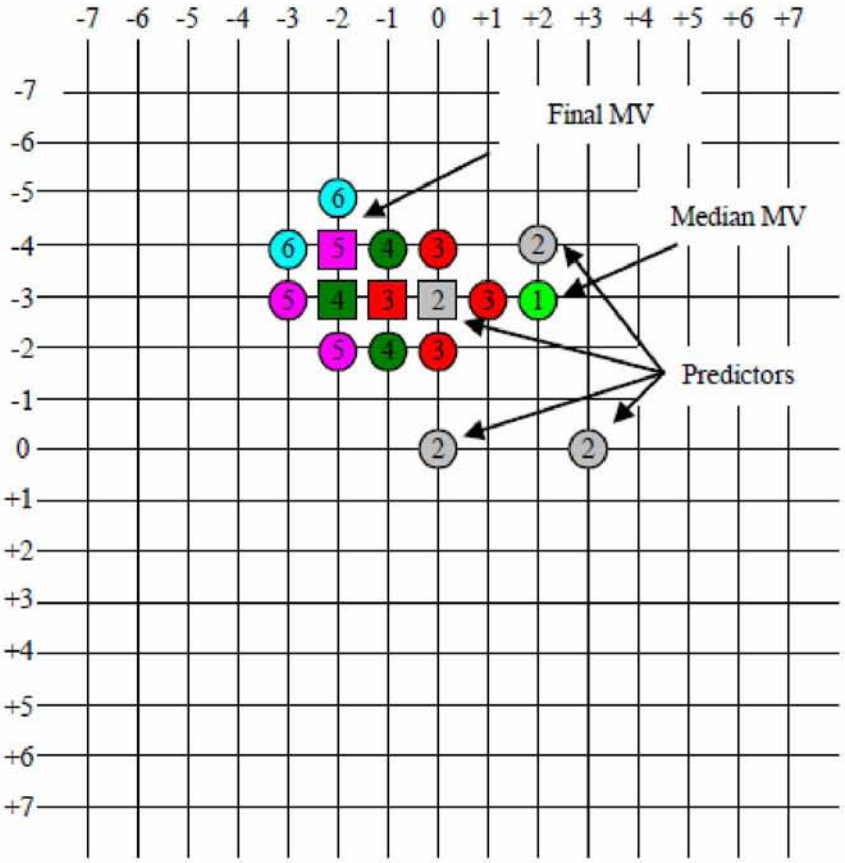
predictors(δηλαδή τις τιμές εκείνες που χρησιμοποιεί για να προβλέψει την θέση της βέλτιστης επιλογής) χρησιμοποιεί και τον μέσο (median predictor), και τους χρονικούς υποψήφιους (temporal candidates), επιπλέον εισάγονται κάποια νέα κατώφλια και η επιλογή ανάμεσα σε μικρό ή μεγάλο σχήμα διαμαντιού γίνεται με άλλο τρόπο.

Γενικά ο αλγόριθμος είναι πιο πολύπλοκος από τον MVFAST με σκοπό να βρίσκει γρήγορα τα διανύσματα κίνησης από κάποιες τετριμμένες περιπτώσεις βίντεο. Επιπλέον εισάγει και νέους predictors όπως το διάνυσμα κίνησης του MB που εξετάσαμε στο προηγούμενο καρέ,μιάς και αναμένω πως στις περισσότερες των περιπτώσεων σύμφωνα με τη συνέχεια του βίντεο, όπου βρήκαμε τη βέλτιστη επιλογή του MB (0,0) στο προηγούμενο καρέ, κάπου κοντά να βρω και τη βέλτιστη για το τρέχον. Αυτός όμως είναι και ο κύριος λόγος για τον οποίο δεν μπορούμε να το χρησιμοποιήσουμε στον αλγόριθμό μας, μιας και για λόγους εξοικονόμησης πόρων επεξεργαζόμαστε τα καρέ ανά δύο.

Predictors:

1. Το διάνυσμα κίνησης του ακριβώς πάνω macroblock
2. Το διάνυσμα κίνησης του ακριβώς αριστερά macroblock
3. Το διάνυσμα κίνησης του πάνω-αριστερά macroblock
4. Υπολογίζεται ένα διάνυσμα κίνησης με συντεταγμένες (i,j) όπου το i είναι ο μέσος όρος των i των τριών παραπάνω predictors και το j ο μέσος όρος των j των τριών παραπάνω predictors, το οποίο λέγεται Median predictor.
5. Το διάνυσμα κίνησης που βρήκαμε σαν βέλτιστη επιλογή για το αντίστοιχο MB του προηγούμενου καρέ.

Συγκεκριμένα ο κώδικάς μας εξετάζει δύο καρέ μεταξύ τους και όχι ολόκληρο βίντεο έτσι δεν μπορούμε να χρησιμοποιήσουμε τα διανύσματα κίνησης από προηγούμενα καρέ, άρα ο PMFAST δεν προσδίδει πολλές νέες βελτιώσεις ενώ εισάγει μεγαλύτερη πολυπλοκότητα στο κώδικα. Έτσι θα προτιμήσουμε τον MVFAST.



Σχήμα 12: Παράδειγμα εφαρμογής του PMVFAST, τα τετράγωνα blocks αναπαριστούν το ελάχιστο σε κάθε επανάληψη.

Sequence	Format	BR	SA		Motion Estimation Algorithms				
					FS	*PMVFAST*	TSS	NTSS	DS
Hall monitor	QCIF	10	16	PSNR	30.35	*30.34*	29.79	29.79	30.32
				Speed Up	1	*378*	41	55	77
Mother & Daughter	QCIF	24	16	PSNR	34.80	*34.78*	34.81	34.78	34.78
				Speed Up	1	*311*	41	54	74
Coastguard	QCIF	48	16	PSNR	28.88	*28.9*	28.77	28.75	28.73
				Speed Up	1	*158*	41	41	58
Coastguard	CIF	112	16	PSNR	27.03	*27.14*	26.71	26.66	26.44
				Speed Up	1	*114*	41	37	49
			32	PSNR	27.06	*27.19*	26.72	26.69	26.46
				Speed Up	1	*431*	156	142	187
Foreman	CIF	112	16	PSNR	30.04	*29.95*	29.46	29.54	29.58
				Speed Up	1	*106*	41	39	43
			32	PSNR	30.37	*30.24*	29.52	29.58	29.63
				Speed Up	1	*386*	156	147	158
Foreman	CIF	1024	16	PSNR	35.47	*35.54*	34.79	34.92	34.97
				Speed Up	1	*203*	41	44	55
			32	PSNR	35.53	*35.59*	34.79	34.91	34.97
				Speed Up	1	*763*	156	169	208
		512	16	PSNR	34.51	*34.56*	33.88	34.02	34.07
				Speed Up	1	*139*	41	40	47
			32	PSNR	34.84	*34.87*	33.90	34.02	34.09
				Speed Up	1	*526*	156	153	173
Table tennis	SIF	1024	16	PSNR	34.98	*34.98*	34.71	34.82	34.92
				Speed Up	1	*310*	41	50	68
			32	PSNR	35.00	*34.97*	34.72	34.83	34.90
				Speed Up	1	*1165*	155	191	258

Σχήμα 13: Για κάθε βίντεο, βλέπουμε το μέγεθος του,το bitrate του, την εμβέλεια αναζήτησης (search area), την ποιότητα σε db, και βελτίωση σε σχέση με τον αλγόριθμο Πλήρους Αναζήτησης full search. Βλέπω εύκολα, πόσο καλύτερο είναι σε απόδοση από όλους τους άλλους.

Τελική επιλογή αλγόριθμου εκτίμησης κίνησης

Από την παραπάνω έρευνα στους αλγόριθμους για εκτίμηση κίνησης και συνυπολογίζοντας ότι αυτό που χρειαζόμαστε για την υλοποίηση μας είναι ένας πολύ γρήγορος αλγόριθμος με καλή ποιότητα και αξιοπιστία, κρίνει την επιλογή του MVFASTμονόδρομο. Αυτό μπορούμε να το πούμε μιας και είναι μαζί με τον PMVFAST οι πιο γρήγοροι και από τους γρήγορους, οι πιο αξιόπιστοι. Ο μόνος πιο αξιόπιστος είναι ο Πλήρους Αναζήτησης (Full Search) ο οποίος φυσικά είναι απαγορευτικός λόγω της χαμηλής ταχύτητάς του.

Τέλος ανάμεσά τους επιλέγω τον MVFAST λόγω απλότητας του σε σχέση με τον PMVFAST αλλά και ανάγκης για εκτέλεση του κώδικα σε πραγματικό χρόνο, μιας και όπως είπαμε

παραπάνω η αποθήκευση των διανυσμάτων κίνησης των προηγούμενων καρέ θεωρείτε υπερβολή από άποψη πόρων και χρόνου

Θέματα με την υλοποίηση

Κατά την υλοποίηση του αλγόριθμου για εκτίμηση κίνησης, έχουμε το εξής πρόβλημα: Όταν ο αλγόριθμος θα έχει ένα macroblock στις άκρες του τρέχοντος καρέ, θα πρέπει να ψάξει γύρω από το καρέ αναφοράς, για να βρει τη βέλτιστη επιλογή. Επειδή όμως είναι στο άκρο του καρέ, τα όρια του καρέ είναι λογικό να περιορίζουν το ψάξιμο του αλγόριθμου.

Στα περισσότερα βίντεο, παρατηρούμε ότι στα άκρα τους εξαφανίζονται και εμφανίζονται συνέχεια νέα αντικείμενα, με αποτέλεσμα να χάνετε η βέλτιστη επιλογή για ένα αντικείμενο που εξαφανίζεται και αυτό έχει σαν αποτέλεσμα ο αλγόριθμος να βρίσκει μέτριες επιλογές σε αυτά τα MB.

Μία έξυπνη λύση, είναι να επεκτείνουμε το καρέ προς τα άκρα που συνορεύει, αντιγράφοντας στα νέα pixel που θα εισαχθούν τις τιμές που έχουν τα ακραία pixel. Με άλλα λόγια επεκτείνω το καρέ μου, με ότι είχε στα όρια του έως τώρα, έτσι θα δημιουργήσω νέα MB τα οποία θα έχουν ότι έχει απομείνει από το αντικείμενο πριν εξαφανιστεί και αυτό αναμένω να μου δώσει μία σαφώς καλύτερη επιλογή από ότι προηγουμένως.

Σαφώς όμως αυτή η διαδικασία καταναλώνει χρόνο ο οποίος αργότερα μπορεί να καταστεί πολύτιμος, έτσι φροντίζουμε να κάνουμε την επέκταση ορίων (boundary extension) επιλογή του χρήστη.

Κεφάλαιο 4ο: Αναγνώριση Κινούμενων Αντικειμένων & Διαχωρισμός

Η Πληροφορία της Διασποράς Υπολειπόμενων Pixels

Σημαντικό ρόλο στον κώδικα μου για την αναγνώριση των αντικειμένων θα παίξει η διασπορά υπολειπόμενων pixels. Όπως γνωρίζουμε διασπορά ενός MB είναι πόσο απέχουν οι τιμές των pixels μεταξύ τους, ή αλλιώς πόσο απέχουν από τον μέσο όρο τους. Από που προέρχεται αυτή η διασπορά όμως;

Η διασπορά που μας ενδιαφέρει και θα δώσει όλη την απαραίτητη πληροφορία για τον χωρισμό των macroblock (MB), είναι η διασπορά της διαφοράς του MB βέλτιστη επιλογή που βρήκαμε στο τρέχον καρέ, με το αρχικό MB στο καρέ αναφοράς. Δηλαδή έστω ότι έχουμε ένα MB στο τρέχον καρέ, με τον αλγόριθμο MVFAST θα ψάξουμε να βρούμε πιο MB του καρέ αναφοράς έχει τις κοντινότερες τιμές σε αυτό, δηλαδή το μικρότερο SAD. Όταν το βρούμε αυτό το macroblock, δεν περιμένουμε να είναι όλες οι τιμές τους ίδιες, εκτός φυσικά από εξαιρετικές

περιπτώσεις, έτσι, θα το αφαιρέσουμε από το αρχικό MB (ένα προς ένα pixel η αφαίρεση) και θα σχηματίσουμε ένα νέο MB το οποίο θα περιέχει τις διαφορές αυτών των δύο. Η διασπορά αυτού του MB υπολειπόμενων pixels είναι αυτή που χρειαζόμαστε. Τι αντιπροσωπεύει όμως αυτή η διασπορά;

Έστω ότι για ένα MB αυτή η διασπορά είναι υψηλή, τι μπορεί να σημαίνει αυτό; Αν ένα MB έχει μεγάλη διασπορά στο MB διαφοράς, αυτό σημαίνει πως κάποια pixels του έχουν υψηλή τιμή και κάποια άλλα χαμηλή άρα σε κάποια σημεία του MB υπάρχει έντονη διαφοροποίηση, άρα προφανώς δεν μπορούμε πλέον να μιλάμε για ένα αντικείμενο.

Το πρώτο πράγμα που δείχνει η μεγάλη διασπορά σε ένα MB υπολειπόμενων pixels είναι φυσικά έντονη κίνηση, γιατί η μεγάλη διασπορά σημαίνει μεγάλη απόσταση των pixels από τη μέση τιμή, που με τη σειρά του σημαίνει ότι η βέλτιστη επιλογή που βρήκαμε διαφέρει αρκετά από το MB στο καρέ αναφοράς και άρα είχαμε έντονη αλλαγή στη κίνηση.

Δεύτερον όμως, μεγάλη διασπορά μπορεί να σημαίνει ότι τα μισά pixels έχουν μικρή τιμή και τα άλλα μισά μεγάλη, δηλαδή με άλλα λόγια μιλάμε για *διαφορετικό αντικείμενο*. Αυτός είναι και ένας τρόπος να κρίνουμε αν ένα MB αποτυπώνει πάνω από ένα αντικείμενα.

Το τι κριτήριο θα χρησιμοποιήσουμε όμως για να διακριτοποιήσουμε μια υψηλή διασπορά σε μεγάλη κίνηση ή διαφορετικά αντικείμενα θα φανεί παρακάτω, στην διαδικασία του διαχωρισμού (splitting).

gebardas@uth.gr

Σχήμα 14: Στις δύο παραπάνω εικόνες παρατηρούμε με πιο σκούρο χρώμα τα macroblocks 16x16 με υψηλή μη μηδενική διασπορά (πάνω από 2).

Η Πληροφορία του Διανύσματος Κίνησης

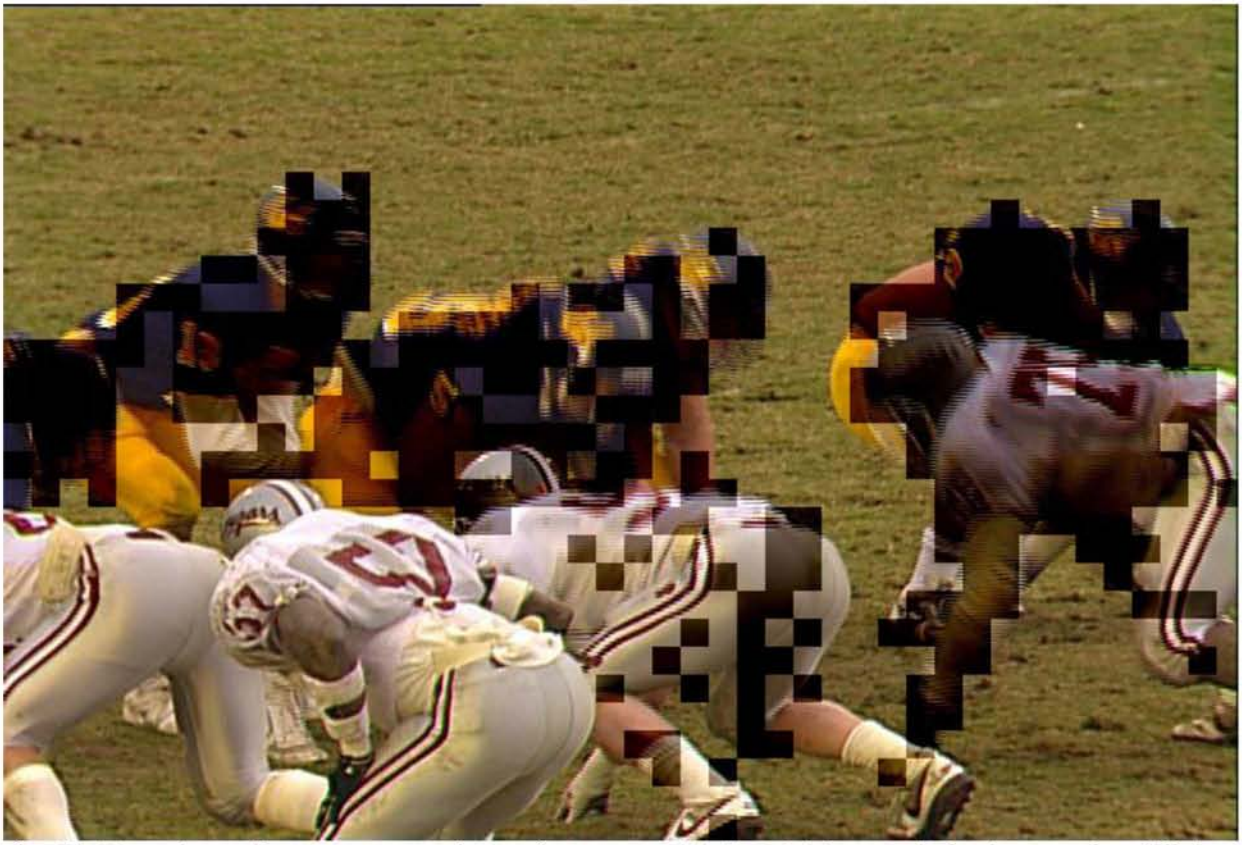
Όπως είπαμε και παραπάνω, το διάνυσμα κίνησης δεν είναι τίποτα παραπάνω από την διαφορά της θέσης του MB στο τρέχον καρέ με το MB του καρέ αναφοράς που βρήκαμε σαν βέλτιστη επιλογή (best match). Δηλαδή στην ουσία όπως λέει το όνομά του, είναι το διάνυσμα της κίνησης, που δείχνει δηλαδή τι κίνηση έκανε το συγκεκριμένο MB στο χρόνο που πέρασε ανάμεσα στη λήψη των δύο αυτών καρέ.

Εδώ τα πράγματα είναι πιο ξεκάθαρα όσων αφορά του τί μας δείχνει το διάνυσμα κίνησης. Όπως είπαμε, μας δείχνει το πώς κινήθηκε ένα MB, άρα τα πράγματα είναι ξεκάθαρα, αν ένα MB έχει διάνυσμα κίνησης μηδέν, δηλαδή (0,0), τότε είναι ακίνητο άρα είναι φόντο. Αν τώρα έχει μη μηδενικό διάνυσμα κίνησης, κοιτάμε και τα γύρω MB και αναμένουμε όσα έχουν το ίδιο διάνυσμα κίνησης να οριοθετούν ένα (το ίδιο) αντικείμενο. Διαφορετικά δύο διαφορετικά γειτονικά διανύσματα κίνησης(π.χ. το (-1,4) και το (4,-1)) δείχνουν διαφορετική κίνηση άρα διαφορετικό αντικείμενο.

Άρα, παίρνουμε από το διάνυσμα κίνησης ποια macroblocks κινούνται και από τη διασπορά, ποια macroblocks είναι οριακά, δηλαδή εκεί που αλλάζει το αντικείμενο. Είναι όμως το

16x16 macroblock αρκετό για να σχηματίσει ένα αντικείμενο; Μάλλον όχι,γιατί αντιπροσωπεύει μικρή λεπτομέρεια και θα χρειαστούμε περισσότερη για τα άκρα των αντικειμένων. Έτσι χρειαζόμαστε αρχικά μία τεχνική η οποία θα σπάει τα macroblocks με μεγάλη διασπορά υπολειπόμενων pixels σε μικρότερα κομμάτια και αργότερα μία ακόμη που θα ενώνει τα όμοια macroblocks, δηλαδή αυτά που αναφέρονται στο ίδιο αντικείμενο. Όπως θα φανεί και κάτω στις εικόνες, η συμβολή και των δύο μεγεθών είναι απαραίτητη, μιας και κανένα από τα δύο δεν θα έδινε από μόνο του ρεαλιστικά αποτελέσματα.

gebardas@uth.gr



Σχήμα 15: Στα σχήματα έχουμε με πιο σκούρο χρώμα τα macroblocks 16x16 τα οποία δεν έχουν μηδενικό διάνυσμα κίνησης. Μπορούμε έτσι άμεσα να συγκρίνουμε τη πληροφορία που μας δίνει η διασπορά και το διάνυσμα κίνησης. Φαίνεται εύκολα πως το διάνυσμα κίνησης μας δίνει καθαρά ποια MB κινούνται ενώ η διασπορά μας δίνει τα όρια ανάμεσα σε δύο αντικείμενα π.χ σε ένα αντικείμενο που κινείται και το φόντο.

Χωρισμός των macroblocks (Splitting)

Splitting με κριτήριο τη διασπορά υπολειπόμενων pixels (Variance Splitting)

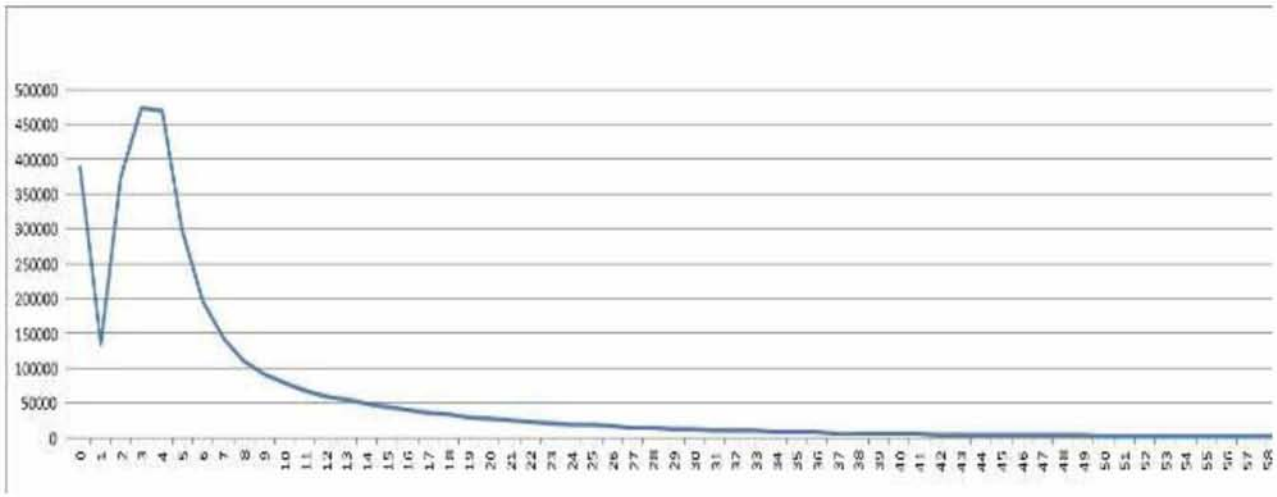
Όπως είπαμε και παραπάνω, ο χωρισμός των macroblocks σε μικρότερα είναι επιτακτικός μιας και το 16x16 MB δεν δίνει ρεαλιστικά σχήματα στα αντικείμενά μας.

Το επόμενο ζήτημα που θα μας απασχολήσει είναι πότε θα χωρίζουμε ένα macroblock. Είπαμε παραπάνω πως θα το κάνουμε όταν βλέπουμε πως η διασπορά του είναι υψηλή, πρέπει όμως να βρούμε την χρυσή τομή στο μέγεθος της διασποράς από το οποίο και πάνω θα θεωρούμε ότι το macroblock αναφέρεται σε δύο αντικείμενα και άρα πρέπει να χωριστεί.

Τα πιθανά κριτήρια για διαχωρισμό μπορούνε να είναι:

- Μαζεύουμε στατιστικά για πολλά MB, πολλών καρέ και πολλών βίντεο ποικίλου περιεχομένου, φτιάχνουμε μια κατανομή με πόσα MB αντιστοιχούν σε κάθε τιμή από διασπορά, έτσι βρίσκουμε ένα ασφαλή αριθμό για την διασπορά πάνω από τον οποίο θεωρούμε ότι το MB πρέπει να διαχωριστεί(tracing). Στην συνέχεια κάθε MB που βρίσκουμε να έχει διασπορά μεγαλύτερη ή ίση με τον αριθμό που βρήκαμε το σπάμε στα τέσσερα,υπολογίζουμε εκ νέου τις διασπορές των τεσσάρων νέων MB και σπάμε ξανά όσα συνεχίζουν να έχουν μεγάλη διασπορά μέχρι ελάχιστο macroblock μήκος και ύψος = 4 pixels. Το να σπάσουμε σε πιο μικρά κομμάτια είναι υπερβολή μιας και πλέον δεν θα κάνει διαφορά στο μάτι, απλά θα χάσουμε χρόνο στην εκτέλεση.
- Βρίσκουμε τη διασπορά των τεσσάρων προς διαχώριση MB (ξεχωριστά το κάθε ένα) χωρίς να έχουμε εφαρμόσει διαχωρισμό ακόμη και αν αυτή έχει μειωθεί αισθητά σε σχέση με την αρχική (ολόκληρου του αρχικού MB) σημαίνει ότι αξίζει να χωρισθεί αλλιώς όχι.

Επιλέγουμε τον διαχωρισμό με κατώφλι για εξοικονόμηση χρόνου και μαζεύοντας στατιστικά από όλα τα MB όλων των καρέ όλων των βίντεο έχουμε την κατανομή:



Σχήμα 16: Πολλά από τα βίντεο μας έχουν καρέ που δεν αλλάζουν (δύο ή και παραπάνω ακίνητα καρέ συνεχόμενα) άρα βρίσκουμε πολλά με διασπορά μηδέν. Αν αγνοήσουμε τη μηδενική διασπορά μπορούμε να παρατηρήσουμε ότι ένα ρεαλιστικό κατώφλι είναι κάπου στο 20-22.

Κοιτώντας την κατανομή και υπολογίζοντας την διασπορά της κατανομής, μπορούμε να συμπεράνουμε πως ένα ασφαλές και ρεαλιστικό κατώφλι είναι το 20. Έτσι για κάθε MB με διασπορά ίση ή μεγαλύτερη του 20, το χωρίζουμε σε τέσσερα μικρότερα MB.

Διαχωρισμός με υπολογισμό μερικών sad (Partial Sad Splitting)

Όταν υπολογίζουμε το sum of absolut difference, υπάρχει μια λεπτομέρεια που αξίζει να παρατηρήσουμε. Αντί να υπολογίζουμε κατευθείαν το sad του 16x16 MB είναι το ίδιο να υπολογίζουμε τέσσερα sad, αυτά των τεσσάρων τεταρτημορίων(8x8 pixels) του MB και μετά απλά να τα αθροίζουμε για να βγάλουμε το sad ολόκληρου του macroblock.

Αυτό πρακτικά είναι το ίδιο από άποψης πόρων και χρόνου εκτέλεσης με πριν αλλά μας δίνει ένα ισχυρό κριτήριο για διαχωρισμό, μιας και αντιλαμβανόμαστε πως αν ένα sad από αυτά διαφέρει πολύ από τα άλλα(4 φορές μεγαλύτερο για παράδειγμα), αυτό σημαίνει πως σε εκείνο το κομμάτι του block υπάρχει κάτι που διαφέρει σε σχέση με όλο το υπόλοιπο macroblock. Αυτό δεν είναι τίποτα άλλο από ένα διαφορετικό αντικείμενο ή και φόντο.

Όπως και το παραπάνω κριτήριο έτσι και αυτό δεν χρειάζεται πολύ επεξεργαστικό χρόνο και αυτό το καθιστά εξαιρετικά ισχυρό στην επιλογή για τον διαχωρισμό των blocks.

Σχήμα 17: Τα τέσσερα partial sads 8x8 που θα υπολογιστούν αντί του ενός 16x16 sad

Επιλογή τελικού κριτηρίου διαχωρισμού

Για την υλοποίησή μας θα χρειαστεί να διαλέξουμε μεταξύ αυτών των δύο κριτηρίων. Τρέχοντας πολλά παραδείγματα για τα δύο κριτήρια, παρατηρούμε πως αυτά διαφέρουν κυρίως σε ευαισθησία. Δηλαδή το κριτήριο της διασποράς έτσι θα το ονομάζουμε από εδώ και πέρα, για κατώφλι 20, βρίσκει τα πιο προφανή macroblocks για διαχωρισμό και αγνοεί κάποια λιγότερο προφανή. Από την άλλη το κριτήριο των μερικών sad (με κατώφλι 4,δηλαδή το μεγαλύτερο μερικό sad να είναι τέσσερεις φορές μεγαλύτερο από το μικρότερο) βρίσκει όλα εκείνα τα macroblocks

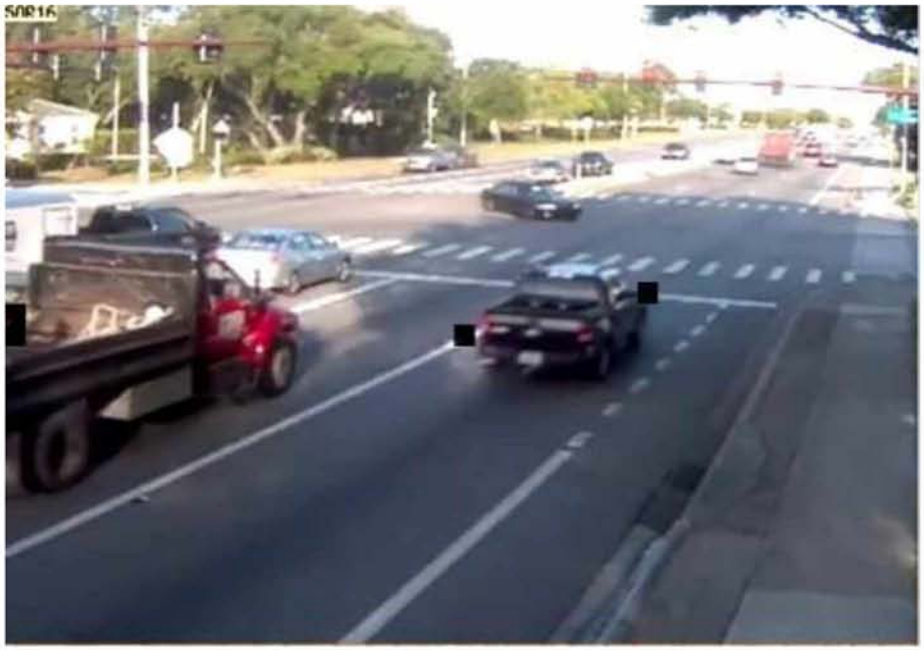
που χρίζουν προσοχής, αλλά σε βίντεο που τραβήχτηκαν υπό κακές καιρικές συνθήκες ή που είναι χαμηλής ποιότητας (παρατηρείται δηλαδή έντονος θόρυβος) χωρίζει εκτός των σωστών και όλα εκείνα τα MB που παρατηρείται πολύ θόρυβος.

Για αυτό το λόγο επιλέχθηκε ένα συνδυαστικό κριτήριο, δηλαδή για να χωριστεί ένα block θα πρέπει να ικανοποιούνται *και τα δύο κριτήρια*, με χαμηλότερες τιμές κατωφλιών σε αυτά(8 για το κριτήριο των μερικών sad και 30 για το κριτήριο της διασποράς).

Σχήμα 18: Παράδειγμα διαχωρισμού των MB με το κριτήριο των διασπορών,σε βίντεο με άψογη ποιότητα, για κατώφλι=20. Με μαύρο είναι τα blocks που το κριτήριο επισημαίνει ότι πρέπει να διαχωριστούν. Επιλέγει τα πιο προφανή MB που περιέχουν και αντικείμενα και φόντο, αγνοεί σχεδόν τις σκιές.

gebardas@uth.gr

Σχήμα 19: Παράδειγμα διαχωρισμού των MB με το κριτήριο των μερικών sad,σε βίντεο με άψογη ποιότητα, για κατώφλι=4.Με μαύρο είναι τα blocks που το κριτήριο επισημαίνει ότι πρέπει να διαχωριστούν. Κυρίως επιλέγει τα MB που περιέχουν και αντικείμενο και φόντο, βρίσκει ακόμη και τις σκιές.

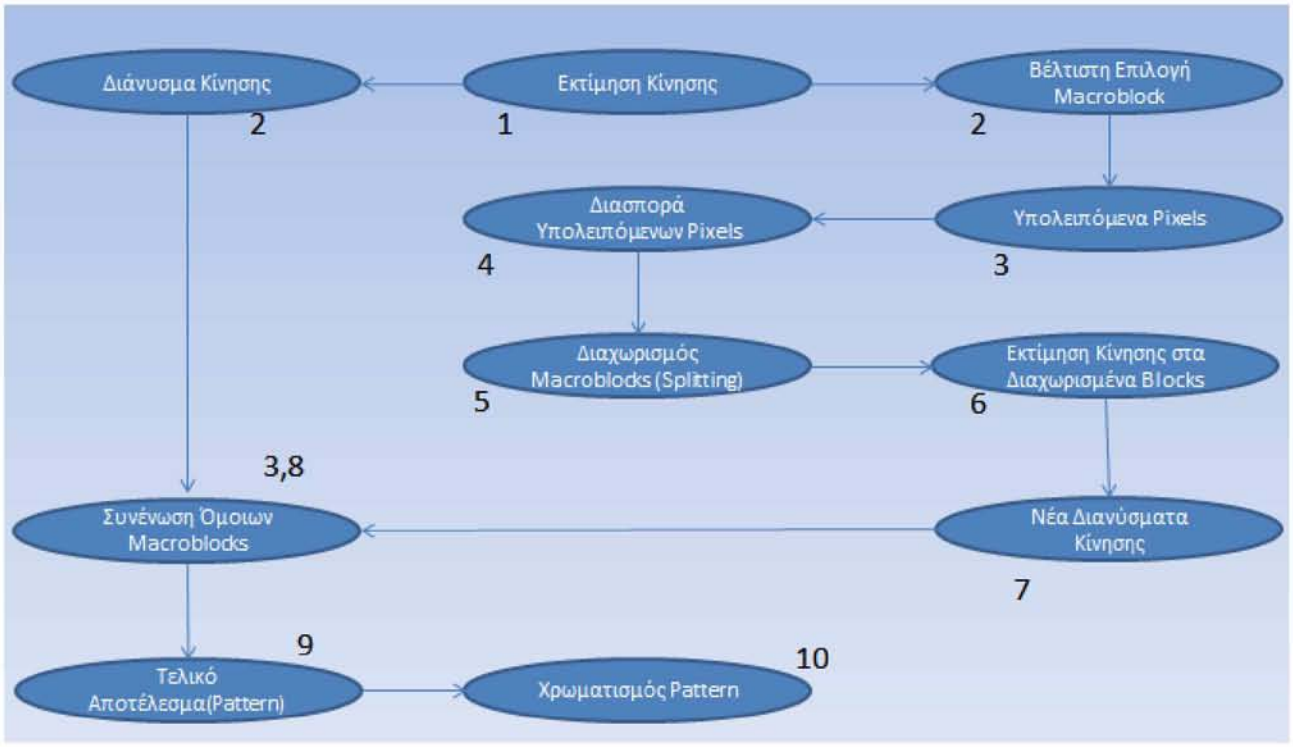


Σχήμα 20: Παράδειγμα του κριτηρίου των διασπορών σε ένα πραγματικό βίντεο από κάμερα κυκλοφορίας. Παρατηρώ ότι από όλες τις κινήσεις μόνο 2 MB στο μαύρο όχημα θεωρεί ότι χρειάζονται διαχωρισμό. Άρα χαμηλή ευαισθησία.

gebardas@uth.gr

Σχήμα 21: Το ίδιο βίντεο για το κριτήριο των μερικών sad, βλέπουμε ότι είναι υπερευαίσθητο, μιας και όλα τα MB στα οποία συναντά θόρυβο λόγω της χαμηλής ποιότητας του βίντεο, θεωρεί ότι πρέπει να διαχωριστούν. Άρα υψηλή ευαισθησία.

gebardas@uth.gr

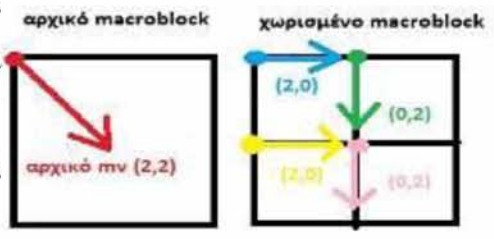


Σχήμα 22: Η διαδικασία που ακολουθείται κατά την εκτέλεση του αλγορίθμου.

Κεφάλαιο 5ο:Συνένωση διαχωρισμένων blocks

Η Διαδικασία της Συνένωσης (Merging Process)

Σε αυτό το κεφάλαιο θα ασχοληθούμε με τη διαδικασία της συνένωσης των blocks(Merging). Όπως είπαμε και παραπάνω μετά τον διαχωρισμό των blocks χρειαζόμαστε να τα ενώσουμε (όσα αναφέρονται στο ίδιο αντικείμενο πάντα) για να καταλήξουμε και στο τελικό αντικείμενο.



Σχήμα 23: Χωρισμός των macroblock

Πλέον, όσα από τα macroblocks μας είχαν μεγάλη διασπορά έχουν χωριστεί και άρα η διασπορά τους έχει μειωθεί. Έτσι, αυτό που χρειαζόμαστε είναι να συνενώσουμε τα διάφορα MB για να φτιάξουμε το pattern δηλαδή το σχήμα γύρω από το αναγνωρισμένο

gebardas@uth.gr

αντικείμενο.

Για να το κάνουμε αυτό πρέπει να χρησιμοποιήσουμε κάποια κριτήρια με την βοήθεια των οποίων θα πρέπει να βγάλουμε το συμπέρασμα ότι δύο blocks αναφέρονται στο ίδιο αντικείμενο και άρα πρέπει να τα ενώσουμε. Πότε όμως δύο γειτονικά blocks αναφέρονται στο ίδιο αντικείμενο;

Ένα αξιόπιστο κριτήριο είναι όταν φυσικά τα δύο blocks έχουν το ίδιο διάνυσμα κίνησης, αυτό ισχύει γιατί σίγουρα δύο γειτονικά blocks που έχουν το ίδιο διάνυσμα κίνησης αναφέρονται στο ίδιο αντικείμενο(εφόσον έχουν διαχωριστεί, δηλαδή και τα τέσσερα μερικά sad τους είναι παρόμοια και δεν έχουν υψηλή διασπορά).

Εδώ όμως πρέπει να προσέξουμε ότι, όταν κάναμε τον χωρισμό των MB, δεν δώσαμε στα νέα σπασμένα blocks διαφορετικό διάνυσμα κίνησης, αλλά αυτά κράτησαν το διάνυσμα κίνησης που είχε ο πρόγονος τους, το διαχωρισμένο MB δηλαδή. Έτσι αν χρησιμοποιήσουμε μόνο αυτό το κριτήριο θα συνενωθούν τα ίδια MB που πριν είχαν χωρισθεί και έτσι δεν αλλάζει τίποτα.

Έτσι, καταλήγουμε φυσικά, πως πρέπει να τρέξουμε εκ νέου τον αλγόριθμο mvfast για τα χωρισμένα blocks που δημιουργήθηκαν, για να πάρουν αυτά το νέο διάνυσμα κίνησης τους.

Όταν το νέο διάνυσμα κίνησής τους είναι έτοιμο, τότε μπορούμε να το χρησιμοποιήσουμε για να κρίνουμε εάν ανήκουν ή όχι στο ίδιο αντικείμενο. Ο αλγόριθμος προσπελαύνει τα blocks σαν 4x4, αρχίζοντας πάντα, από κάποιο που δεν έχει επισκεφθεί προηγουμένως και κοιτάζει τα 4 γειτονικά του, το πάνω, το κάτω, το δεξιά και το αριστερά blocks, αν βρει ότι κάποιο από αυτά έχει ίδιο ή παρόμοιο διάνυσμα κίνησης(αναλόγως το κατώφλι συνενώσεως που έχει δώσει ο χρήστης,δηλαδή για κατώφλι ίσο με 2, το (1,1) θα συνενωθεί με το (2,2) γιατί το άθροισμα των διαφορών των δύο συντελεστών είναι ίσο με 2) το συνενώνει και συνεχίζει από εκείνο, αναδρομικά, μέχρι να προσπελαστούν όλα τα blocks του καρέ. Για το κριτήριο συνένωσης έχουν γίνει δύο υλοποιήσεις:

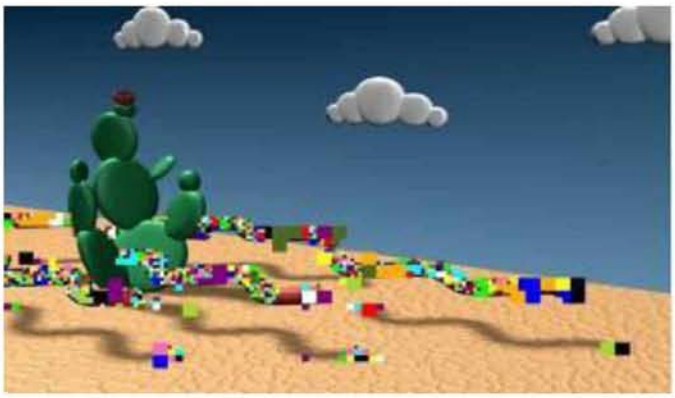
1. Αλγόριθμος Μερικής Συνένωσης (Συνένωση μόνο σε κινούμενα blocks): Η πρώτη ιδέα ήταν πως πρέπει να ξεχωρίσουμε από την αρχή τα blocks που κινούνται από αυτά που είναι ακίνητα. Αυτό θα μας έδινε εύκολα με τη μία κάποια περιγράμματα (patterns), μιας και όσα blocks δεν κινούνται ,όταν ενωθούν σχηματίζουν όπως είναι λογικό και τα αντικείμενα που κινούνται, πριν καν γίνει συνένωση σε αυτά. Στη συνέχεια πραγματοποιείται και συνένωση στα κινούμενα patterns, με κριτήριο το διάνυσμα κίνησής τους, κοιτώντας τα γειτονικά blocks και συνενώνοντας μόνο εάν η διαφορά του διανύσματος κίνησης τους είναι μικρότερη από ένα κατώφλι που δίνει σαν όρισμα ο χρήστης(κατώφλι συνενώσεως). Έτσι

σχηματίζονται και τα αντικείμενα μέσα στα patterns που κινούνται. Εδώ, αν ο χρήστης δώσει μια μεγάλη τιμή κατωφλίου, τότε θα συνενώσει όλα τα κινούμενα αντικείμενα και απλά θα ξεχωρίζει τα κινούμενα από τα μη.

2. Αλγόριθμος Ολικής Συνένωσης (Συνένωση σε όλα τα blocks): Εδώ η υλοποίηση δεν ξεχωρίζει κινούμενα από μη κινούμενα blocks και εκτελείται αναδρομικά κάθε φορά για κάθε block, κοιτάζοντας δεξιά,πάνω,κάτω και αριστερά για blocks με παρόμοιο(όπως το ορίσαμε παραπάνω) διάνυσμα κίνησης, αν βρει κάποιο τέτοιο τα συνενώνει και τρέχει αναδρομικά από αυτό. Αυτό έχει σαν αποτέλεσμα πως για ένα μεγάλο κατώφλι, θεωρητικά άπειρο, ο αλγόριθμος θα συνενώσει όλα τα blocks, με αποτέλεσμα να δημιουργήσει ένα αντικείμενο στην ουσία, όλο το καρέ.

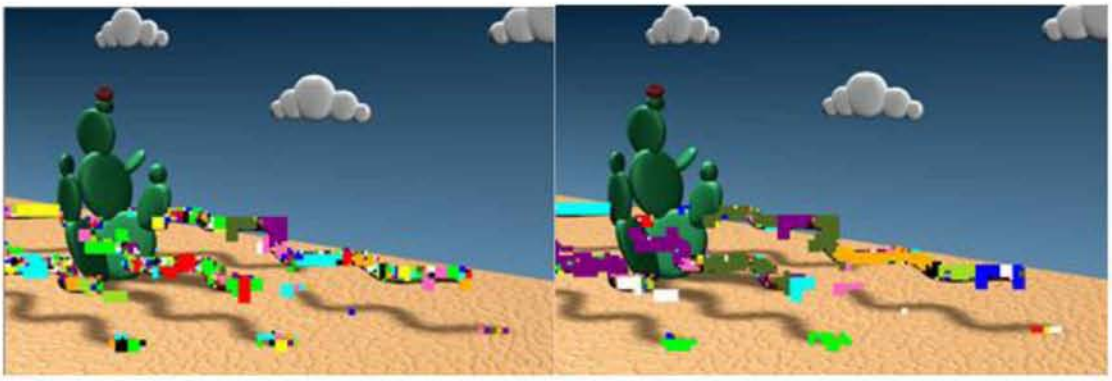
Οι δύο αυτοί αλγόριθμοι είναι υλοποιημένοι και είναι στην κρίση του χρήστη ποιόν θα χρησιμοποιήσει. Μετά από πολλές εφαρμογές, παρατηρήθηκε πως ο πρώτος δουλεύει καλύτερα σε καθαρά και στατικά βίντεο (δηλαδή που η λήψη είναι 100% ακίνητη), ενώ ο δεύτερος δουλεύει καλύτερα σε βίντεο με θόρυβο ή που η λήψη τους έγινε με κίνηση.

Σχήμα 24: Το αρχικό καρέ του βίντεο πριν την επεξεργασία, το οποίο αποτελεί ένα πολύ δύσκολο βίντεο, μιας και τα φιδάκια κινούνται και κάτω και δεξιά, αλλά έχουν και μπερδεμένα χρώματα.

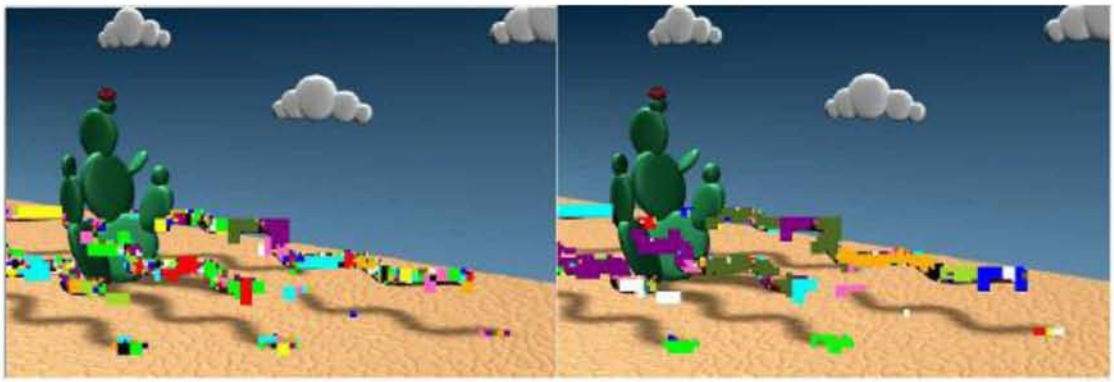


Σχήμα 25: Ο αλγόριθμος που δεν ξεχωρίζει κινούμενα από ακίνητα αντικείμενα (αριστερά, για κατώφλι=2-συνενώνει για διαφορά<=2) παρατηρείται ότι δίνει πολύ άσχημα αποτελέσματα, μιας και αναγνωρίζει πολλά μικρά αντικείμενα. Αντίθετα για κατώφλι 5 τα αποτελέσματα είναι καλύτερα

.

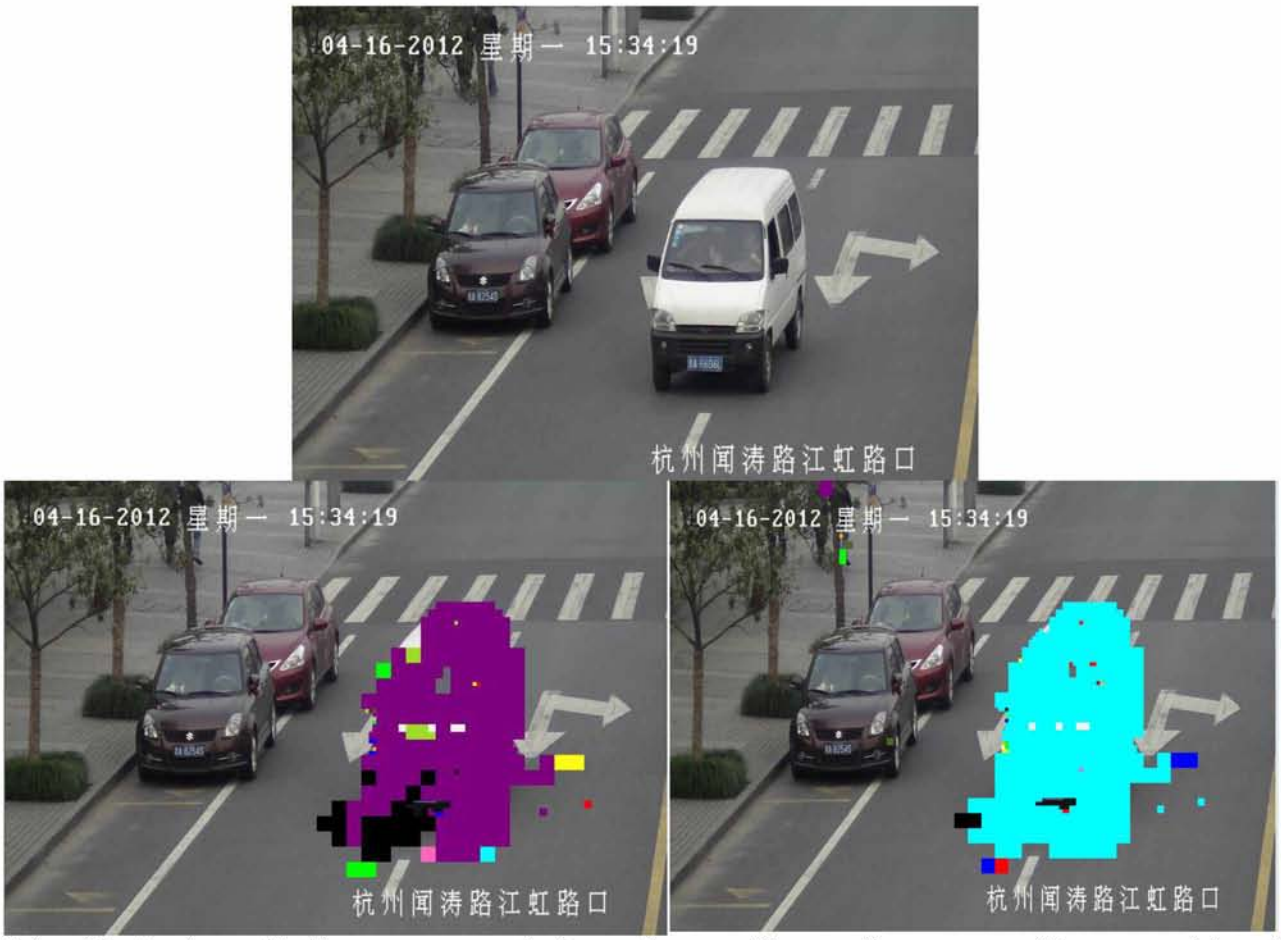


Σχήμα 26: Ο αλγόριθμος ολικής συνένωσης (αριστερά για κατώφλι 2,συνενώνει για διαφορά <=2) παρατηρείται ότι δίνει πολύ άσχημα αποτελέσματα, γιατί δίνει πολλά μικρά αντικείμενα. Αντίθετα για κατώφλι 5 (δεξιά) τα αποτελέσματα είναι καλύτερα.

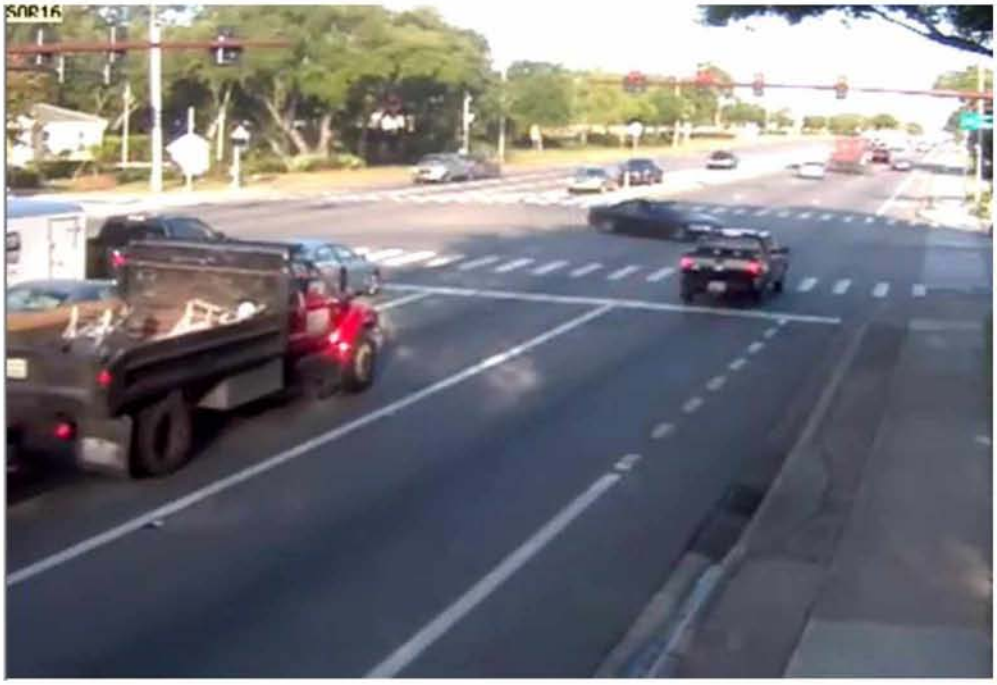


Σχήμα 27: Παρόμοια αποτελέσματα δίνει και ο αλγόριθμος μερικής συνένωσης, για τιμή κατωφλίου 2 αριστερά και σαφώς καλύτερα για τιμή 5 δεξιά.

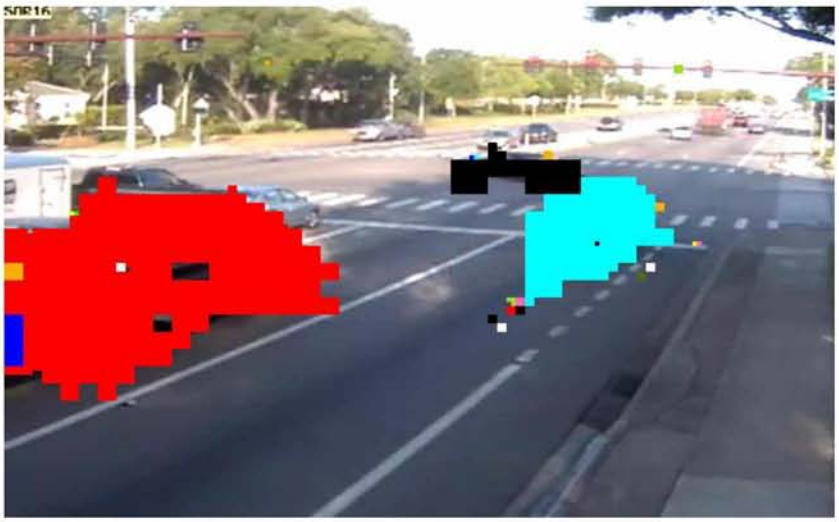
Σχήμα 28: Για κατώφλι όμως 10, ο παραπάνω αλγόριθμος (μερικής συνένωσης) βγάζει εξαιρετικά αποτελέσματα, μιας και αναγνωρίζει ακριβώς όλα τα αντικείμενα και τις σκιές τους. Εδώ λοιπόν φαίνεται η υπεροχή του αλγορίθμου που ξεχωρίζει το φόντο, μιάς και για υψηλότερή τιμή κατωφλίου συνενώνει τα στοιχείου του κινούμενου αντικειμένου και δίνει σωστά αποτελέσματα, ενώ ο αλγόριθμος ολικής συνένωσης για μεγάλο κατώφλι συνενώνει και με το φόντο που δίνει λανθασμένα αποτελέσματα.



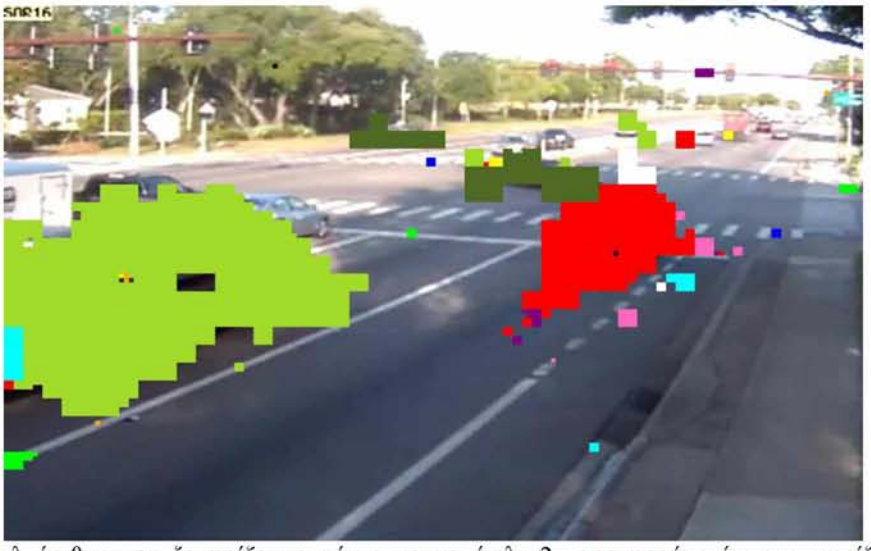
Σχήμα 29: Για ένα απλό βίντεο με γραμμική κίνηση, έχουν το ίδιο αποτέλεσμα και οι δύο για χαμηλή τιμή κατωφλίου(2). Παρατηρείστε πως ο αλγόριθμος ολικής συνένωσης (κάτω αριστερά) αγνοεί τα άτομα στην πάνω αριστερά γωνία του καρέ, μιας και τα συνενώνει με το φόντο, ενώ ο μερικής συνένωσης (κάτω δεξιά) τα αναγνωρίζει κανονικά επειδή βρίσκει κίνηση εκεί.



Σχήμα 31: Ένα δύσκολο βίντεο με πολλές κινήσεις, θόρυβο, και πολλά αμάξια που κινούνται αντίθετα.



Σχήμα 30: Με τον αλγόριθμο που δεν ξεχωρίζει το φόντο, και κατώφλι=2, παρατηρούμε τρία κύρια αντικείμενα, το σημαντικό σημείο είναι πως δεν ένωσε τα δύο αντικείμενα πάνω γιατί εμφανώς έχουν διαφορετικά διανύσματα κίνησης και σωστά τα ξεχωρίζει σαν δύο διαφορετικά αντικείμενα.



Σχήμα 32:Με τον αλγόριθμο που ξεχωρίζει το φόντο,για κατώφλι=2 παρατηρούμε ότι αναγνωρίζονται περισσότερα αντικείμενα αλλά συνεχίζουν να αναγνωρίζονται τα 2 διαφορετικά οχήματα.

Σχήμα 33:Για κατώφλι=10 και τον ίδιο αλγόριθμο, πλέον τα αντικείμενα αναγνωρίζονται σαν ένα και συνεπώς ο αλγόριθμος αποτυγχάνει.

Έτσι, συμπεραίνουμε πως στα περισσότερα βίντεο ειδικά για τιμές κατωφλίου υψηλότερες από 5 ο αλγόριθμος μερικής συνένωσης έχει σαφές προβάδισμα, αλλά το γεγονός πως βρίσκει όλα τα αντικείμενα με κίνηση δεν είναι πάντα θετικό, ειδικά σε βίντεο με υψηλά ποσοστά θορύβου.

Κεφάλαιο 6ο:Αναγνώριση και χρωματισμός αντικειμένων

Χρωματισμός αντικειμένων σε διαδοχικά καρέ

Όπως είδαμε μέχρι τώρα, διακρίνουμε τα διάφορα αντικείμενα μεταξύ τους με ένα χρώμα πάνω από το εκάστοτε αντικείμενο. Μέχρι τώρα αυτά τα χρώματα ήταν τυχαία.

Αναγνωρίζουμε όμως την ανάγκη, ένα αντικείμενο να έχει το ίδιο χρώμα σε όλη τη διάρκεια ζωής του στο βίντεο(ή του αντικειμένου,μιάς και μπορεί να εξαφανιστεί από το βίντεο). Με αυτό τον τρόπο θα μπορούμε να το ξεχωρίζουμε και να το παρατηρήσουμε σε όλη τη διάρκεια του βίντεο. Πώς θα το πετύχουμε αυτό όμως;

Πλέον πρέπει να γίνει κατανοητό, πως σε αυτό το σημείο, δεν μιλάμε για blocks πλέον, μιας και αυτά έχουν επεξεργαστεί και συνενωθεί σε αντικείμενα που είναι μια τελείως διαφορετική οντότητα.

Για τα αντικείμενα λοιπόν, πλέον χρειαζόμαστε ένα τρόπο να προσεγγίσουμε την θέση τους στο επόμενο καρέ, έτσι ώστε να μπορούμε να τα συνδέουμε στα διάφορα καρέ και να τα "χρωματίζουμε" με το ίδιο χρώμα. Για να το κάνουμε αυτό θα πρέπει να κρατάμε όλη την πληροφορία σχετικά με τα αντικείμενα του τρέχοντος καρέ. Για το κάθε αντικείμενο έχουμε τις εξής πληροφορίες:

- Συντεταγμένες Περιγεγραμμένου Ορθογωνίου (Bounding Box Coordinates): Είναι οι συντεταγμένες του εξωτερικού ορθογωνίου του αντικειμένου.
- Μέγεθος Περιγεγραμμένου Ορθογωνίου (Bounding Box Size): Το μέγεθος σε pixel του περιγεγραμμένου ορθογωνίου.
- Μέγεθος Αντικειμένου (Object Size): Το μέγεθος του αντικειμένου σε pixel.
- Μέσο Διάνυσμα Κίνησης (Average Motion Vectors): Μέσο διάνυσμα κίνησης οριζόντια και κάθετα.
- Χρώμα Αντικειμένου (Object Color): Το χρώμα του αντικειμένου στο προηγούμενο καρέ

Για κάθε καρέ κρατάμε αυτές τις πληροφορίες, για όλα του τα αντικείμενα και με αυτές θα χρησιμοποιήσουμε έναν αλγόριθμο για να βρούμε στο επόμενο καρέ που θα είναι το τρέχον αντικείμενο, για να του δώσουμε το χρώμα που είχε και στο προηγούμενο καρέ.

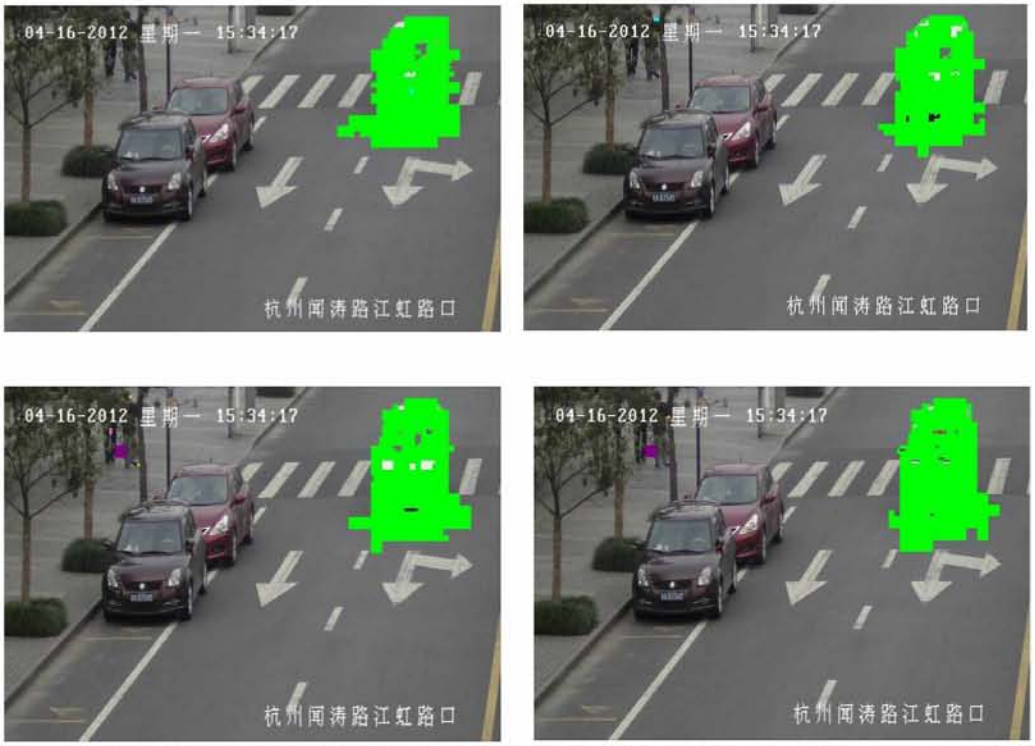
Η λογική του αλγορίθμου είναι απλή:

Έστω ένα αντικείμενο στο καρέ δύο, για να βρούμε σε ποιο αντικείμενο του καρέ ένα αντιστοιχεί, θα ψάξουμε όλα τα αντικείμενα του καρέ ένα, κοιτώντας αν οι συντεταγμένες του τρέχοντος αντικειμένου (συντεταγμένες περιγεγραμμένου ορθογωνίου) πλήν(κατά απόλυτη τιμή) το μέσο διάνυσμα κίνησής του, απέχουν (μία προς μία) απόσταση, το πολύ ίση με ένα κατώφλι που έχει επιλέξει ο χρήστης. Εάν ναι, τότε αυτό το αντικείμενο είναι πιθανό να είναι το ίδιο στο προηγούμενο καρέ, τότε συγκρίνουμε και τα μέσα διανύσματα κίνησής τους(εκτός από το πρώτο καρέ που είναι σαφές ότι δεν έχει διανύσματα κίνησης) και αν είναι ίσα, τότε "χρωματίζουμε" το αντικείμενο στο καρέ δύο με το χρώμα του αντικειμένου στο καρέ ένα. Εάν τώρα, υπάρχουν πάνω από ένα υποψήφια όμοια αντικείμενα, υπολογίζω την τομή των περιγεγραμμένων ορθογωνίων τους, με το περιγεγραμμένο ορθογώνιο του προηγούμενου καρέ και επιλέγω εκείνο που δίνει την μεγαλύτερη τομή, δηλαδή αυτό που μοιάζει περισσότερο.

Με άλλα λόγια, βασιζόμαστε στη λογική πως ένα αντικείμενο στο επόμενο καρέ αναμένουμε να έχει μετακινηθεί περίπου όσο και στο προηγούμενο, δηλαδή περίπου όσο λέει το μέσο διάνυσμα κίνησής του.

Εδώ κάποιος θα σκεφτεί γιατί ο αλγόριθμος να μην τα συγκρίνει και ως προς το μέγεθός

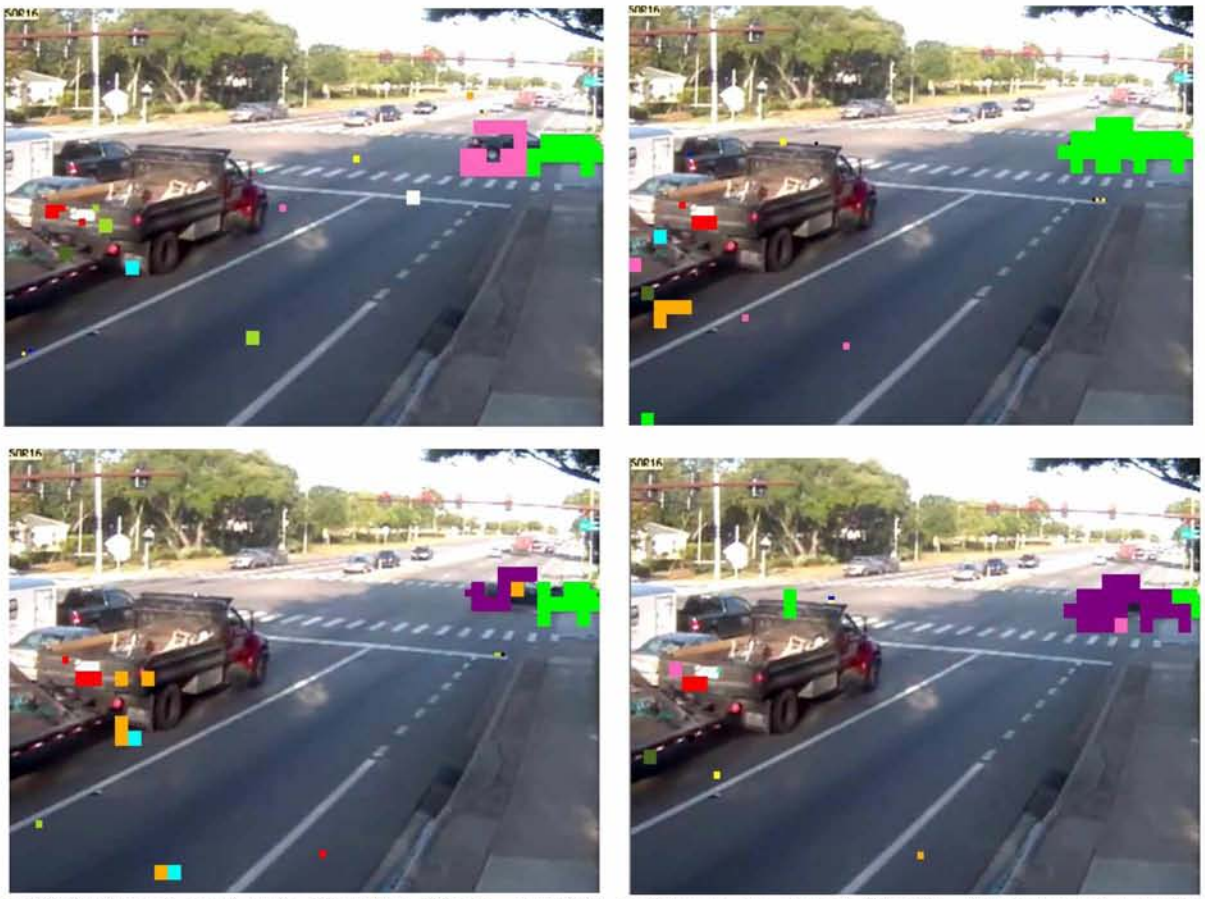
τους, μιας και το ίδιο αντικείμενο αναμένουμε να έχει το ίδιο μέγεθος σε δύο διαδοχικά καρέ. Μετά από πολλές εφαρμογές σε βίντεο χαμηλής σχετικά ποιότητας, παρατηρήθηκε ότι η υλοποίησή μας, σε ένα καρέ μπορεί να αναγνωρίζει ορθά ένα αντικείμενο, αλλά στο επόμενο, μπορεί να το χωρίσει στα δύο ή να το ενώσει με κάποιο άλλο γειτονικό ή και με τη σκιά του, λόγω θορύβου που παρατηρείται ενδιάμεσα, έτσι αποφασίστηκε πως δεν είναι αξιόπιστο κριτήριο για τον αλγόριθμο επιλογής χρώματος.



Σχήμα 34:Διαδοχικά καρέ ενός βίντεο από κάμερα υψηλής ποιότητας που δεν κινείται. Ο αλγόριθμος βρίσκει εύκολα το χρώμα ακόμη και για πολύ χαμηλή τιμή κατωφλίου(20). Τα κενά που φαίνονται ανάμεσα στο αντικείμενο γίνονται επειδή σε εκείνα τα σημεία ο αλγόριθμος δεν βρίσκει κίνηση το οποίο θα δούμε παρακάτω.

Ο αλγόριθμος φροντίζει, όταν δεν βρει καμία επιλογή να διαλέγει ένα χρώμα τυχαία. Αυτό έχει σαν αποτέλεσμα σε βίντεο χαμηλής ποιότητας ο χρήστης να πρέπει να αυξήσει το κατώφλι για αυτή τη σύγκριση, γιατί είναι προτιμότερο να βρει κάποια κακή επιλογή, παρά να επιλέξει τυχαία.

Έτσι γίνεται φανερό πως η επιλογή του κατωφλίου από το χρήστη εδώ, όπως και στα περισσότερα κατώφλια άλλωστε, πρέπει να είναι τέτοια ώστε ούτε να επιλέγεται συνέχεια τυχαίο χρώμα, λόγω μη εύρεσης του σωστού αντικειμένου, αλλά ούτε να διαλέγει ένα λάθος χρώμα, πιθανώς από κάποιο γειτονικό αντικείμενο που μοιάζει, μιας και σε πολλές περιπτώσεις, ειδικά στην εφαρμογή σε φανάρια, θα εμφανίζονται αντικείμενα από το πουθενά αλλά θα εξαφανίζονται κιόλας. Τότε προτιμούμε να δοθεί σε ένα τέτοιο αντικείμενο ένα τυχαίο χρώμα και όχι κάποιο γειτονικό, το οποίο θα μπερδέψει το μάτι του παρατηρητή.



Σχήμα 35:Διαδοχικά καρέ ενός δύσκολου βίντεο, χαμηλής ποιότητας, με αρκετό θόρυβο. Αναγκαστικά ο χρήστης πρέπει να αυξήσει το κατώφλι, για να βρίσκει αντιστοιχίες για όλα τα αντικείμενα, αλλιώς θα κάνει τυχαία επιλογή χρώματος, το οποίο είναι χειρότερο. Έτσι παρατηρούμε ότι ο αλγόριθμος δυσκολεύεται να βρει σωστά το χρώμα των δύο οχημάτων πάνω δεξιά στο καρέ και κάνει λάθη στην επιλογή του. Ο θόρυβος φαίνεται από τα μεμονωμένα χρώματα σε φαινομενικά ακίνητα σημεία, όπως ο δρόμος για παράδειγμα.

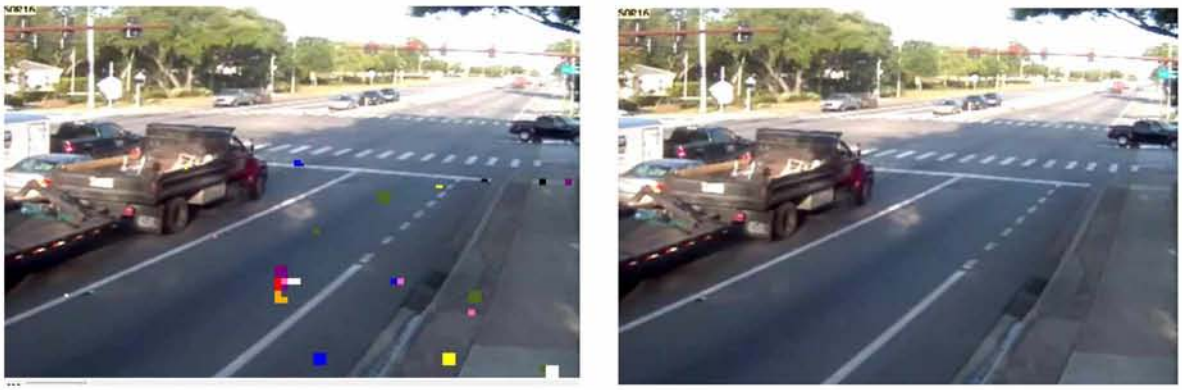
Αναγνώριση ατελειών στο βίντεο και διόρθωση

Σε αυτό το σημείο έχουμε εφαρμόσει όλους τους αλγόριθμους, η διάσπαση, η συνένωση και ο χρωματισμός των αντικειμένων στα διαδοχικά καρέ έχουν γίνει επιτυχώς, παρόλα αυτά υπάρχουν κάποιες ατέλειες σε βίντεο που εφαρμόστηκε ο αλγόριθμος, που είναι είτε προϊόν θορύβου είτε απορρέει των διαδικασιών της υλοποίησής μας. Τέτοιες ατέλειες μπορεί να είναι:

- *Αντικείμενα μικρού μεγέθους* που δημιουργούνται και εξαφανίζονται μέσα σε ένα καρέ, είναι *αποτέλεσμα θορύβου* στο βίντεο και παρατηρούνται πιο εύκολα σε σημεία όπου δεν υπάρχει κίνηση. Ακόμη και τα καλύτερης ποιότητας βίντεο παρατηρείται να έχουν τέτοια σημεία

και η υλοποίησή μας πρέπει να τα αντιμετωπίζει.

Ένας καλός τρόπος είναι η εφαρμογή ενός κατωφλίου κατά τον χρωματισμό, το οποίο αναφέρεται στο μέγεθος του αντικειμένου σε pixel και το χρωματίζει μόνο εάν είναι μεγαλύτερο αυτού.



Σχήμα 36:Φαίνεται η εφαρμογή της υλοποίησης σε ένα ακίνητο καρέ. Αριστερά φαίνονται τα διάφορα αντικείμενα και δεξιά πώς αυτά αποκόπτονται με την χρήση ενός ασφαλούς κατωφλίου 512 pixel. Δηλαδή ο αλγόριθμος «χρωματίζει» ένα αντικείμενο, μόνο αν είναι πάνω από 512 pixel.

- Η αλλαγή σκηνής, είναι κάτι που δεν θα δούμε συχνά σε κάμερες κυκλοφορίας, αλλά όπου συναντάτε είναι ενοχλητική, μιας και η μέθοδός μας βρίσκει αλλαγές παντού στο καρέ και τα χρωματίζει όλα σαν κίνηση. Μία καλή λύση, είναι η εφαρμογή ενός κατωφλίου στην μέση διασπορά του καρέ η οποία εάν ξεπερνά μία ασφαλής τιμή (10 είναι η αρχική επιλογή μας) τότε αγνοείται όλο το καρέ και δεν χρωματίζεται τίποτα.
- Κενά μέσα σε ένα ενιαίο αντικείμενο, μπορεί να δημιουργούνται κυρίως είτε γιατί υπάρχει χαμηλή κίνηση, είτε γιατί το αντικείμενο έχει σε πολλά σημεία ένα ενιαίο χρώμα ή και τα δύο. Αυτό δυσκολεύει τον αλγόριθμο να αναγνωρίσει την κίνηση και έτσι δημιουργούνται αυτά τα κενά. Μία καλή λύση είναι αυτή της Διαστολής (Dilation), δηλαδή να χρωματίσουμε τις άκρες των αντικειμένων μας με ένα πιο χοντρό πινέλο ή αλλιώς την επέκταση του χρώματος του αντικειμένου προς τα έξω.

Σχήμα 37:Φαίνεται η εφαρμογή διαστολής στην υλοποίησή μας. Αρχικά παρατηρείται ότι υπάρχουν πολλά κενά στο όχημα και με την εφαρμογή διαστολής το πρόβλημα διορθώθηκε χωρίς το αντικείμενο να αλλάξει σχήμα ή να χαλάσει κάτι γύρω του, άρα είναι μια αποδεκτή λύση.

Κεφάλαιο 7ο:Θεωρητικά και πρακτικά προβλήματα της υλοποίησης

Χαμηλή κίνηση

Το πρόβλημα

Εφαρμόζοντας την υλοποίησή μας σε πολλά διαφορετικά βίντεο, παρατηρούμε δυσκολία αναγνώρισης των αντικειμένων σε κάποια από αυτά.

Χαρακτηριστικό παράδειγμα είναι τα βίντεο τραβηγμένα από πηγή υψηλής ταχύτητας, δηλαδή από πηγή που απεικονίζει πολλά καρέ ανά δευτερόλεπτο. Αυτά τα βίντεο έχουν το χαρακτηριστικό ότι σε κάθε καρέ παρατηρείται μικρή διαφοροποίηση στη κίνηση και είναι ένα σύνηθες φαινόμενο για τις νέες κάμερες κυκλοφορίας. Έτσι η υλοποίηση μας συναντά δυσκολία να

βρει ολόκληρο το αντικείμενο, μιας και (όπως είδαμε και παραπάνω) αφήνει πολλά κενά ανάμεσα, αφού υπάρχουν σημεία που το αντικείμενο έχει ομοιόμορφο χρώμα και η βέλτιστη επιλογή του αλγορίθμου mvfast μπορεί να βρεθεί οπουδήποτε υπάρχει το ίδιο χρώμα. Αυτό έχει σαν αποτέλεσμα σε πολλά block να μην αναγνωρίζει την κίνηση και εκεί να συναντάμε τα κενά ανάμεσα στα κινητά μέρη του αντικειμένου.

Λύση: Μεγαλύτερο MB μέγεθος

Όπως είπαμε και παραπάνω, ο λόγος που έχουμε αυτό το πρόβλημα σε ορισμένα βίντεο, είναι γιατί η κίνηση είναι χαμηλότερη και τα αντικείμενα σε μερικά τους σημεία έχουν ομοιόμορφο χρώμα, έτσι είναι λογικό πως αυτό το πρόβλημα ισχύει για την τρέχουσα υλοποίηση επειδή το αρχικό MB μέγεθος 16x16 είναι μικρό σε σχέση με τις ομοιομορφίες του χρώματος του αντικειμένου. Για αυτό και αν γίνει η υλοποίηση για μεγαλύτερο μέγεθος MB (32x32, 64x64) τότε αναμένω να έχω κάλυψη σε αυτά τα σημεία και ο αλγόριθμος να μην σπάει εκείνα τα MB μιας και θα τα θεωρεί ακίνητα.

Η επιλογή αυτή όπως κατανοούμε φυσικά, μας κοστίζει σε ταχύτητα όμως, οπότε πρέπει να χρησιμοποιείται μόνο όπου είναι πραγματικά απαραίτητη.

Λύση: Επεξεργασία παραπάνω από ένα καρέ κάθε φορά

Όπως έγινε κατανοητό για την υλοποίησή μας, αυτή κάθε φορά επεξεργάζεται ένα τρέχον καρέ(current) σε σχέση με το προηγούμενο, καρέ αναφοράς (reference). Στο πρόβλημά μας, ανάμεσα σε αυτά τα δύο καρέ συναντάτε πολύ χαμηλή κίνηση. Τι θα γινόταν όμως αν το τρέχον από το καρέ αναφοράς είχαν απόσταση μεγαλύτερη από ένα καρέ;

Με άλλα λόγια λέμε πως θα μπορούσε ο αλγόριθμος να κοιτά τα καρέ ανά δύο ή και παραπάνω, αναλόγως το πόσο χαμηλή είναι η κίνηση που συναντά σε αυτά.

Εισάγουμε δηλαδή ένα βήμα (step) το οποίο προστίθεται σε κάθε # καρέ αναφοράς για να βρεθεί ποιο θα είναι το τρέχον καρέ που θα ασχοληθεί ο αλγόριθμος. Αυτό το βήμα είναι εύκολο να αυτοματοποιηθεί και να το επιλέγει ο αλγόριθμος από μόνος του, υπολογίζοντας απλά ένα μέσο όρο των διανυσμάτων κίνησης στα αντικείμενα που έχουν βρεθεί στα πρώτα καρέ και από τη στιγμή που αυτός είναι χαμηλός να αυξάνει το βήμα.

Το πρόβλημα που δημιουργείται όμως, είναι πως με αυτή την υλοποίηση αναλόγως το βήμα θα μειώνονται και τα καρέ του βίντεο εξόδου, γιατί ο αλγόριθμος όπως είναι κατανοητό

επεξεργάζεται λιγότερα καρέ πλέον. Για παράδειγμα, για βήμα=2 και βίντεο 400 καρέ, ο αλγόριθμος θα δώσει έξοδο 400/2=200 καρέ.

Η λύση σε αυτό το πρόβλημα είναι να αντιγράφουμε στην έξοδο το ίδιο καρέ εξόδου όσες φορές είναι και το βήμα. Έτσι για βήμα=3 θα σώζει τρεις φορές το καρέ εξόδου στο τέλος και λόγο της χαμηλής κίνησης, αυτό δεν θα είναι κάτι το παρατηρήσαμε από το γυμνό μάτι του παρατηρητή.

Κίνηση προς τη κάμερα ή μεγέθυνση αντικειμένου

Υπάρχει το ενδεχόμενο να έχω ένα αντικείμενο, το οποίο κινείται προς τη κάμερα, τότε στην ουσία αυτό το αντικείμενο μεγεθύνεται στο μάτι του παρατηρητή όπως πλησιάζει.

Σε μία τέτοια περίπτωση ο αλγόριθμός μας είναι λογικό να αποτυγχάνει να βρει σωστά το αντικείμενο αυτό. Αναλυτικά ο mvfast θα βρει τις κινήσεις, αλλά θα υποθέσει ότι το αριστερό μέρος του αντικειμένου κινείται προς τα αριστερά και το δεξί προς τα δεξιά, ο διαχωρισμός θα διασπάσει τα MB εκείνα και η συνένωση θα συνενώσει στην ουσία τα άκρα του αντικειμένου, θεωρώντας τα πως είναι ξεχωριστά αντικείμενα.

Έτσι βλέπουμε πως ο αλγόριθμος σε τέτοιες περιπτώσεις αποτυγχάνει.

Στρέψη του αντικειμένου

Όπως και πιο πάνω, έτσι και όταν ένα αντικείμενο στρέφεται, ως προς οποιονδήποτε άξονα του, τότε ο αλγόριθμος αδυνατεί να το αναγνωρίσει.

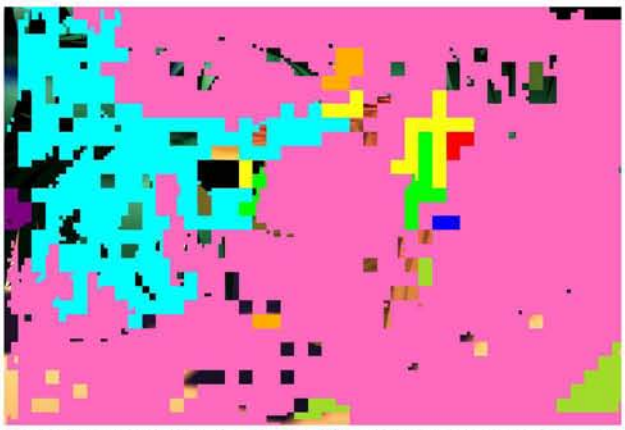
Αυτό γίνεται, γιατί φυσικά δεν κινείται ολόκληρο το αντικείμενο ως προς μία κατεύθυνση, παρά διαφορετικά κομμάτια του κινούνται προς διαφορετικές κατευθύνσεις. Έτσι ο αλγόριθμος της συνένωσης, όπως είναι λογικό θα συνενώσει σαν ένα αντικείμενο μόνο εκείνα τα σημαία με όμοιο διάνυσμα κίνησης και όχι όλο το αντικείμενο.

Έτσι και εδώ ο αλγόριθμός αποτυγχάνει.

Κίνηση της συσκευής λήψης

Αν κινείται όλη η συσκευή λήψης (κάμερα κ.α) τότε τα πράγματα γίνονται ακόμη πιο άσχημα για τον αλγόριθμο. Λόγω της κίνησης αυτής, δεν υπάρχει πλέον φόντο, και άρα κινούνται τα πάντα.

Έτσι η υλοποίησή μας είτε θα δώσει κίνηση σε όλα τα MB και δεν θα καταφέρει να βρει σωστά κάποιο αντικείμενο μιας και όποιο από τους δύο αλγορίθμους συνένωσης και να χρησιμοποιήσουμε, κανείς δεν θα βρει φόντο και θα καταλήξει να χρωματίσει σαν αντικείμενο σχεδόν όλο το καρέ.



Σχήμα 38:Το επεξεργασμένο καρέ του source16 όπου κινείται όλη η κάμερα και αντιμετωπίζεται όλο το καρέ σαν κίνηση.

Συμπεράσματα για την υλοποίηση

Καταλήγουμε λοιπόν έτσι πως η υλοποίησή μας βασίζεται σε κάποιες καθολικές παραδοχές για να βγάλει σωστά αποτελέσματα και να δουλέψει σωστά.

Πρώτα από όλα πρέπει η πηγή λήψης να είναι όσο το δυνατόν ακίνητη,όσο περισσότερο κινείται ή τρέμει εισάγει θόρυβο στα MB και η απόδοσή του μειώνεται. Όταν φυσικά κινείται ολόκληρη η υλοποίηση είναι αδύνατο να λειτουργήσει.

Δεύτερον, όλο το αντικείμενο πρέπει να κινείται προς μια κατεύθυνση για να δουλέψει ο

αλγόριθμος.

Τρίτον, η κίνηση του αντικειμένου επεξεργασίας πρέπει να είναι σταθερή και με μια ικανοποιητική ταχύτητα.

Και τέλος, όσο πιο υψηλής ποιότητας είναι το βίντεο, τόσο καλύτερα και πιο αξιόπιστα τα αποτελέσματα της υλοποίησης μας.

gebardas@uth.gr

Κεφάλαιο 8ο:Αρχιτεκτονική Tensilica Xtensa

Τι πρόβλημα λύνει η αρχιτεκτονική της Tensilica;

Οι επεξεργαστές παραδοσιακά ήταν εξαιρετικά δύσκολο να σχεδιαστούν και να τροποποιηθούν. Γι αυτό τα περισσότερα συστήματα περιέχουν σταθερούς επεξεργαστές οι οποίοι σχεδιάστηκαν για χρήση γενικού σκοπού και μετά ενσωματώθηκαν σε πολλαπλές εφαρμογές στο πέρας του χρόνου. Επειδή όμως αυτοί οι επεξεργαστές είναι γενικού σκοπού, η χρήση τους σε προβλήματα ειδικού σκοπού, δεν ενδείκνυται.

Αν και είναι προτιμότερο να χρησιμοποιήσουμε ένα επεξεργαστή ο οποίος είναι ξεχωριστά κατασκευασμένος να εκτελεί ένα συγκεκριμένο κομμάτι κώδικα καλύτερα (π.χ να τρέχει γρηγορότερα, να καταναλώνει λιγότερη ενέργεια ή να κοστίζει λιγότερο), αυτό σπάνια είναι εφικτό λόγω της δυσκολίας. Ο χρόνος,το κόστος, και το ρίσκο της τροποποίησης ενός υπάρχοντος επεξεργαστή ή σχεδίασης ενός καινούριου είναι πολύ υψηλό.

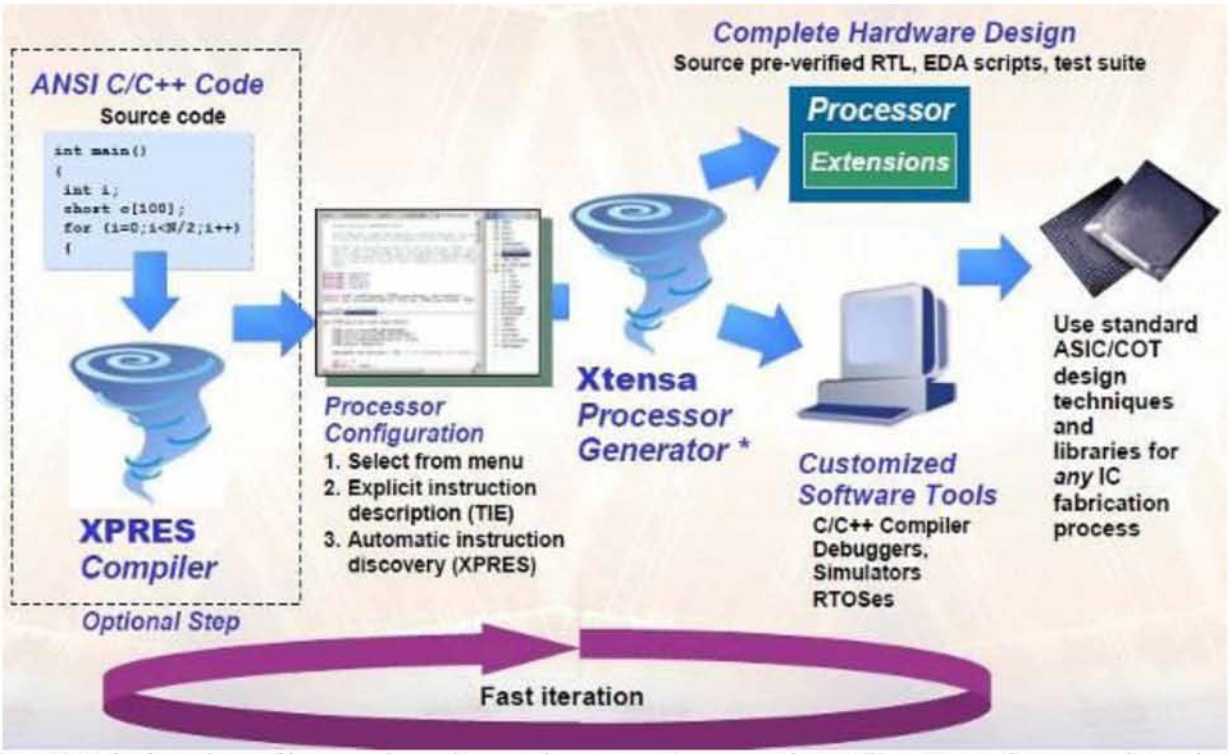
Επιπλέον, δεν είναι πρακτικό να σχεδιάζουμε κλασσικούς επεξεργαστές με περισσότερες δυνατότητες για να καλύπτουν όλες τις εφαρμογές επειδή κάθε συγκεκριμένη εφαρμογή απαιτεί

μόνο ένα συγκεκριμένο σύνολο χαρακτηριστικών. Έτσι ένας επεξεργαστής με χαρακτηριστικά τα οποία δεν χρειάζονται από μια συγκεκριμένη εφαρμογή κοστίζει πολύ και καταναλώνει αχρείαστη ενέργεια, ενώ παράλληλα είναι δύσκολο να γνωρίζουμε όλες τις εφαρμογές που θα τρέξω,προκαταβολικά.

Αν η τροποποίηση επεξεργαστών μπορούσε να αυτοματοποιηθεί και να είναι αξιόπιστη,τότε οι σχεδιαστές συστημάτων θα μπορούσαν να δημιουργήσουν πραγματικά αποδοτικές λύσεις.

Αυτός είναι και ο σκοπός της αρχιτεκτονικής της Tensilica, αφού παρέχει ένα σύνολο τεχνικών και εργαλείων για σχεδίαση μιας λύσης η οποία περιέχει έναν ή παραπάνω επεξεργαστές, κάθε ένας από τους οποίους είναι σχεδιασμένος ειδικά να τρέχει *βέλτιστα μία δεδομένη εφαρμογή*. Το βέλτιστο αυτό τρέξιμο της εφαρμογής μπορεί να αποτελείται από ένα συνδυασμό:

- **Επεκτασιμότητας (Extensibility):** Πρόσθεση νέων αρχιτεκτονικών κομματιών
- **Διαμορφωσιμότητα (Configurability):** Δημιουργία ειδικών διαμορφώσεων στον επεξεργαστή
- **Αναπροσαρμογή hardware (Retargetability):** Εφαρμογή της αρχιτεκτονικής πάνω σε κάποιο hardware για να τρέχει με συγκεκριμένη ταχύτητα, να καταλαμβάνει συγκεκριμένη έκταση και να έχει συγκεκριμένες δυνατότητες σε διαφορετικές εφαρμογές.



Σχήμα 39:Η διαδικασία σχεδίασης ενός επεξεργαστή με την αρχιτεκτονική Tensilica Xtensa όπως περνάει από τον σχεδιαστή και την ίδια την Tensilica

Επεκτασιμότητα: Αναφέρεται στην ικανότητα και δυνατότητα της Tensilica να επεκτείνει την αρχιτεκτονική των επεξεργαστών της με εντολές ειδικευμένες πάνω σε συγκεκριμένες εφαρμογές.

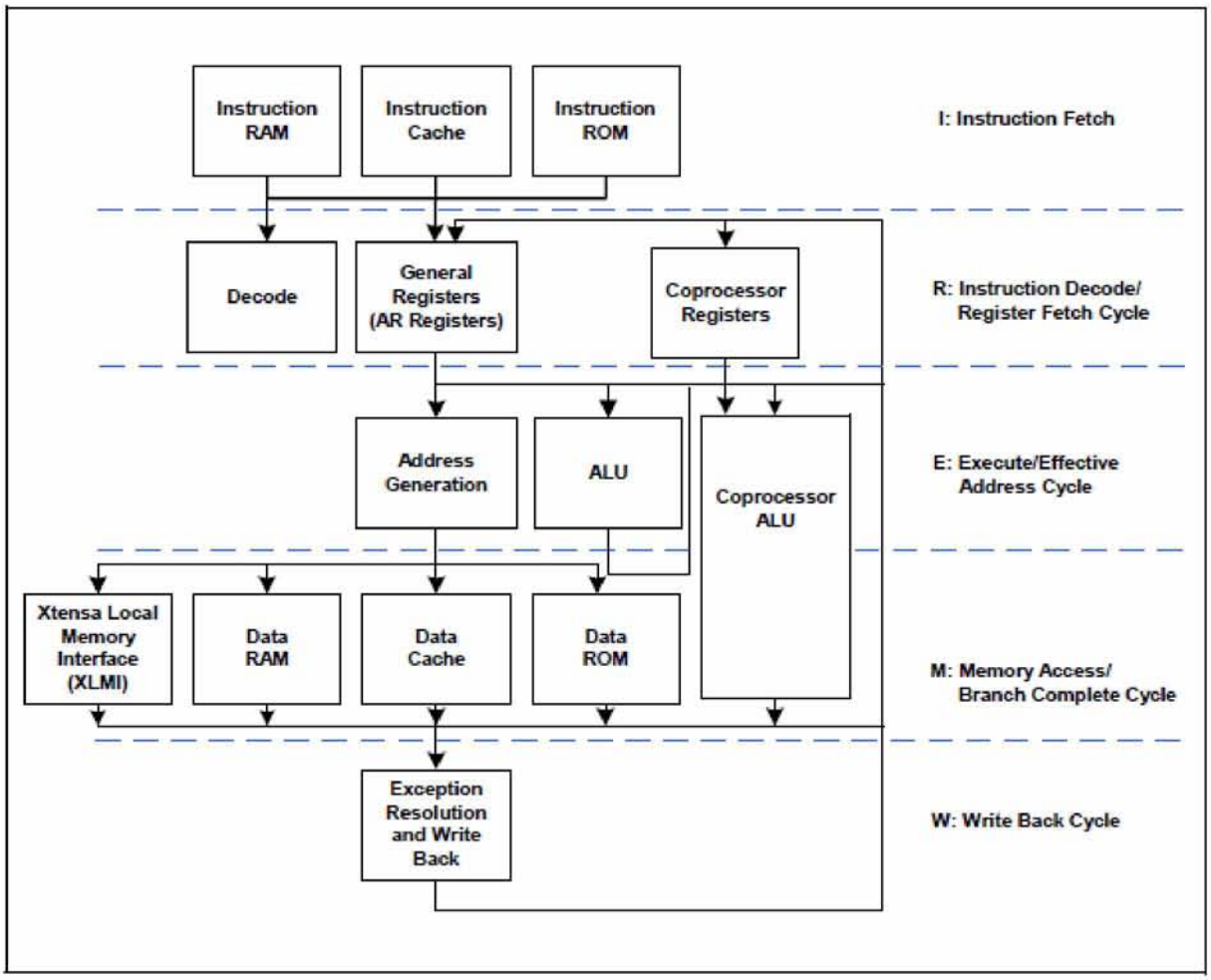
Διαμορφωσιμότητα: Επιτρέπει στον σχεδιαστή του επεξεργαστή να ορίσει αν και πόση προσχεδιασμένη λειτουργικότητα είναι απαραίτητη για ένα συγκεκριμένο προϊόν.

Αναπροσαρμογή hardware: Μετά την σχεδίαση του επεξεργαστή,αυτός πρέπει να υλοποιηθεί σε πραγματικό hardware. Σύνθεση,τοποθέτηση και εργαλεία δρομολόγησης επιτρέπουν υψηλού επιπέδου αναπαραστάσεις να ενσωματωθούν αυτόματα σε πιο λεπτομερείς σχεδιασμούς.

Πλεονεκτήματα

Πλεονεκτήματα της αρχιτεκτονικής:

1. **Πυκνότητα κώδικα:** Οι εντολές στην αρχιτεκτονική Xtensa είναι 24-bit, έτσι το μήκος εντολής και μόνο παρέχει μία άμεση μείωση 25% σε μέγεθος κώδικα, σε σύγκριση με 32-bit ISA. Επιπλέον το Xtensa ISA περιέχει και μια άλλη επιλογή για Πυκνότητα Κώδικα, η οποία προσθέτει 16-bit εντολές οι οποίες ρίχνουν την πυκνότητα κώδικα περαιτέρω.
2. **Χαμηλό Κόστος Υλοποίησης:** Η αρχιτεκτονική Xtensa έχει σχεδιαστεί έτσι, ώστε να παρέχει ελάχιστο κόστος υλοποίησης. Υλοποιείται με pipeline απλών εντολών και direct hardware execution χωρίς microcode. Η βασική αρχιτεκτονική αποφεύγει τις εντολές που θα χρειαστούν επιπλέον πόρτες για διάβασμα ή γράψιμο στον φάκελο καταχωρητών. Αυτό κρατάει τις υλοποιήσεις, χαμηλού κόστους και χαμηλής κατανάλωσης ενέργειας.
3. **Χαμηλή Κατανάλωση Ενέργειας:** Το Xtensa ISA έχει πολλά ενεργειακά βέλτιστα χαρακτηριστικά για τρέξιμο σε συστήματα με μπαταρία. Ο πυρήνας βασίζεται σε 32-bit πράξεις, το οποίο εξοικονομεί ενέργεια. Ο φάκελος καταχωρητών είναι κατασκευασμένος ώστε να εξοικονομεί ενέργεια. Επιπλέον, υπάρχει και το Xtensa Windowed Registers Option, το οποίο εξοικονομεί ενέργεια μειώνοντας τον αριθμό δυναμικών data-memory-references.
4. **Επιδόσεις:** Το Xtensa ISA επιτυγχάνει όλα τα παραπάνω χωρίς να επηρεάζεται η απόδοσή του σε αντίθεση με άλλες αρχιτεκτονικές όπως η ARM ή η MIPS. Επιπλέον παρέχει ένα ολοκληρωμένο σύνολο compare-and-brach εντολών που αυξάνουν την απόδοση, καθώς και εντολές για επαναλήψεις που παρέχουν zero-overhead-looping. Άλλα χαρακτηριστικά της αρχιτεκτονικής ελαχιστοποιούν κρίσιμα μονοπάτια, επιτρέπουν καλύτερη οργάνωση στον compiler και χρειάζονται λιγότερες εντολές για να υλοποιήσουν ένα συγκεκριμένο πρόγραμμα.
5. **Pipeline:** Η αρχιτεκτονική Xtensa μπορεί να υλοποιηθεί μέσα από μία ποικιλία pipelines. Ένα 5-stage load store orientated pipeline είναι το κλασσικό που χρησιμοποιούν πολλοί RISC επεξεργαστές, ενώ παράλληλα υπάρχει και ο 7-stage load store orientated και κάποιοι άλλοι ακόμη.



Σχήμα 40:Το κλασσικό 5-stage load store oriented pipeline

Στάδια Βελτιστοποίησης σε Tensilica Xtensa

- **Καταγραφή των Αποτελεσμάτων (Profiling) της εφαρμογής για εύρεση Σημείων Κατανάλωσης Πολλών Κύκλων (Hotspots):** Κάνω profiling στον κώδικά μου, για να δώ ποια σημεία καταναλώνουν τους περισσότερους κύκλους και άρα τον περισσότερο χρόνο. Με τον τρόπο αυτό θα μπορέσουμε να δούμε που "αργεί" ο κώδικας και βελτιστοποιώντας αυτά τα σημεία θα βελτιστοποιηθεί όλος ο κώδικας.
- **Εισαγωγή TIE's στα Hotspots:** Η κύρια μορφή βελτιστοποίησης στην αρχιτεκτονική Xtensa είναι η εισαγωγή TIE's. Οι TIE's (Tensilica Instruction Extension), είναι μια εντολή η οποία συντάσσεται με γλώσσα περιγραφής hardware

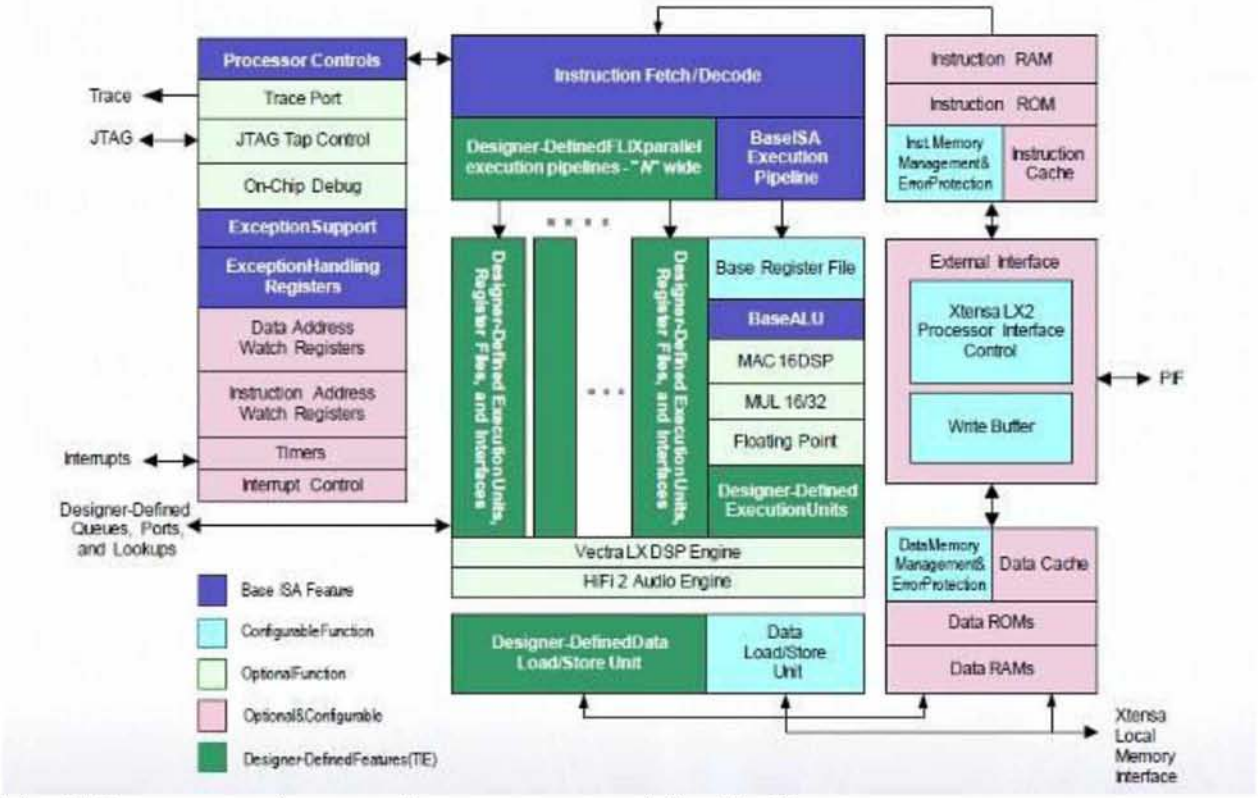
και η οποία εγγυάται από τον μεταγλωττιστή να τρέξει βέλτιστα στον επεξεργαστή που θα σχεδιαστεί.

- **Βελτιστοποίηση των TIE's:** Οι TIE's γενικά έχουν πολλά περιθώρια βελτιστοποίησης μέσα από ένα σύνολο χαρακτηριστικών που εισάγει η αρχιτεκτονική Xtensa.

1. **Fusion:** Αναφέρεται στην ένωση πολλών βασικών πράξεων σε μια εντολή. Ένα παράδειγμα είναι η εντολή Multiply & Accumulate η οποία κάνει ένα πολλαπλασιασμό και μια πρόσθεση. Η τεχνική Fusion είναι πολύ αποδοτική και σημαντική τεχνική για βελτιστοποίηση και μπορεί να χρησιμοποιηθεί πάντα.
2. **SIMD (Μία Εντολή Πολλαπλά Δεδομένα-Signle Instruction Multiple Data):** Με τη βοήθεια νέων φακέλων καταχωρητών μπορούμε να κατασκευάσουμε TIE's που υλοποιούν τη λογική SIMD, δηλαδή μία εντολή η οποία όμως εκτελείται σε πολλά δεδομένα, συνήθως 64 ή128 bit, τα οποία ισοδυναμούν με 8 ή 16 τιμές άρα και τόσες πράξεις.
3. **FLIX:** Η τεχνική FLIX επιτρέπει πολλαπλές πράξεις να εκτελούνται παράλληλα όπου κάθε πράξη είναι ένας πυρήνας Xtensa ή TIE εντολή. Η διαφορά του FLIX με το Fusion είναι πως το Fusion συνδυάζει πολλαπλούς υπολογισμούς σε μια TIE εντολή, ενώ το FLIX επιτρέπει την εκτέλεση πολλαπλών εντολών σε μια. Οι εντολές Fusion απαντώνται κυρίως σε συγκεκριμένες εφαρμογές όπου πολλοί υπολογισμοί συνδυάζονται, ενώ το FLIX είναι γενικό, μιας και κάθε slot μπορεί να περιέχει πολλαπλούς υπολογισμούς τους οποίους ο compiler προσπαθεί να αναθέσει σε μία FLIX εντολή.

- **Μεταγλωττιστής Xpres (Xpres Compiler):** Αναλύει τον κώδικα και κατασκευάζει TIE εντολές οι οποίες τον βελτιστοποιούν αυτόματα.
- **Παραγωγός Εντολών FLIX (FLIX Generator):** Αναλύει τον κώδικα και οργανώνει τις εντολές που υπάρχουν είτε είναι εντολές πυρήνα Xtensa είτε TIE's σε εντολές FLIX οι οποίες βελτιστοποιούν τον κώδικά μας.
- **Σχεδίαση Hardware:** Για να υλοποιηθούν όλες οι παραπάνω τεχνικές πρέπει ο πυρήνας να έχει τα κατάλληλα χαρακτηριστικά. Αυτά πρέπει να οριστούν κατά την κατασκευή του hardware και αναφέρονται στην διαμορφωσιμότητα της αρχιτεκτονικής Xtensa. Τα περισσότερα από αυτά τα χαρακτηριστικά του πυρήνα είναι δυαδικά (binary), δηλαδή ή on ή off, άλλα όμως όχι, όπως π.χ το μέγεθος της

cache κ.ο.κ . Έτσι με τον ορισμό αυτών των χαρακτηριστικών σε επίπεδο hardware, μετά γίνονται διαθέσιμα και σε επίπεδο κώδικα και μπορούμε να τα χρησιμοποιήσουμε στις εντολές TIE. Για παράδειγμα για να χρησιμοποιήσουμε διπλές μονάδες αποθήκευσης/φόρτωσης μνήμης (dual load/store units) πρέπει αυτό να οριστεί στη διαμορφωσιμότητα του πυρήνα, όπως επίσης και το FLIX κ.α .



Σχήμα 41:Ο παραμετροποιήσημος πυρήνας της αρχιτεκτονικής Tensilica Xtensa

Έτσι εφαρμόζοντας τις παραπάνω βελτιστοποιήσεις στον κώδικά μας θα προσπαθήσουμε να τον κάνουμε όσο το δυνατόν ταχύτερο κρατώντας τα ίδια αποτελέσματα φυσικά. Ο λόγος που επιλέχθηκε η αρχιτεκτονική της Tensilica για την βελτιστοποίηση είναι απλός.

Η εργασία μας έχει κύριο σκοπό, όπως είπαμε και παραπάνω, να τρέξει σε φορητές συσκευές (embedded devices) σε πραγματικό χρόνο. Από την περιγραφή της Xtensa αρχιτεκτονικής έγινε σαφές ότι αυτή έχει φτιαχτεί καθαρά για φορητές συσκευές με σκοπό να τρέχει γρήγορα και οικονομικά σε ενέργεια προκαθορισμένα κομμάτια κώδικα. Έτσι με τη βοήθειά της ευελπιστούμε να κατασκευάσουμε ένα πυρήνα που θα ενσωματωθεί σε φορητές συσκευές, θα έχει χαμηλό κόστος, χαμηλή κατανάλωση ενέργειας, μικρή επιφάνεια πυριτίου και θα εκτελεί τον κώδικά μας

πιο γρήγορα από ένα συμβατικό επεξεργαστή γενικού σκοπού.

Όλα τα παραπάνω χαρακτηριστικά της Xtensa αρχιτεκτονικής, την καθιστούν ιδανική αλλά και μοναδική γι αυτό τον σκοπό.

Κεφάλαιο 9ο:Αρχιτεκτονική Tensilica Xtensa-Βελτιστοποιήσεις

Που πρέπει να βελτιστοποιηθεί ο κώδικας;

Αρχικά, για να βελτιστοποιήσουμε τον κώδικά μας, με όποιο τρόπο και αν το κάνουμε αυτό, πρέπει να βρούμε τα hotspots του, δηλαδή τα σημεία εκείνα στα οποία καταναλώνονται πολλοί κύκλοι του επεξεργαστή.

Αυτό αναμένουμε να μας δώσει και σοβαρή βελτίωση στην απόδοση του κώδικα μιας και η βελτιστοποίηση των σημείων που καταναλώνουν πολύ χρόνο είναι και το κλειδί για την απόδοση.

Με τη βοήθεια του Vtune, κάνουμε καταγραφή αποτελεσμάτων (profiling) στον κώδικά μου και παίρνουμε τα εξής αποτελέσματα:

Name	CPU_ sampl	CPU_CLK_U %	CPU_CLK_UN events	Segment	Offset	RVA	Size	Class	Disp
projection3	1,975	31,76%	3,160,000,000	0xFFFFFFFF	0x2570	0x3570	0x2E0		projection3
table_init2	1,307	21,02%	2,091,200,000	0xFFFFFFFF	0x1470	0x2470	0x70		void table_init2(struct mbinfo * *)
sad2	593	9,54%	948,800,000	0xFFFFFFFF	0x310	0x1310	0x2E0		sad2
main	592	9,52%	947,200,000	0xFFFFFFFF	0x33C0	0x43C0	0x90E		main
go_left	316	5,08%	505,600,000	0xFFFFFFFF	0x1650	0x2650	0x1C0		go_left
go_right	291	4,68%	465,600,000	0xFFFFFFFF	0x1810	0x2810	0x1C0		go_right
boundary_extension3	241	3,88%	385,600,000	0xFFFFFFFF	0x60	0x1060	0x200		boundary_extension3
merging7_prep	212	3,41%	339,200,000	0xFFFFFFFF	0x14E0	0x24E0	0x170		void merging7_prep(struct mbinfo
difference2	189	3,04%	302,400,000	0xFFFFFFFF	0xAC0	0x1AC0	0x320		difference2
variance2	177	2,85%	283,200,000	0xFFFFFFFF	0x690	0x1690	0x370		double variance2(short * *,int,int,i
merging9	53	0,85%	84,800,000	0xFFFFFFFF	0x1E90	0x2E90	0x580		merging9
overlap	48	0,77%	76,800,000	0xFFFFFFFF	0x1D40	0x2D40	0x150		overlap
mvfast	45	0,72%	72,000,000	0xFFFFFFFF	0x2850	0x3850	0x380		int mvfast(unsigned char * *,unsig
small_diam	39	0,63%	62,400,000	0xFFFFFFFF	0xF40	0x1F40	0x1F0		small_diam
go_down	35	0,56%	56,000,000	0xFFFFFFFF	0x1B90	0x2B90	0x1B0		void go_down(int,int,int *,int * *,st

- Η συνάρτηση projection3 είναι η συνάρτηση που δημιουργεί το βίντεο εξόδου και την προβολή του βίντεο μετά την επεξεργασία. Αυτή η συνάρτηση παρατηρούμε πως καταναλώνει τους περισσότερους κύκλους, αλλά δεν μας ενδιαφέρει γιατί είναι εκτός της υλοποίησης του κώδικά μας και έχει δημιουργηθεί για λόγους παρουσίασης του αποτελέσματος του κώδικα, άρα απλά την αγνοούμε.
- Η συνάρτηση table_init2 είναι η πρώτη συνάρτηση που ανήκει στον κώδικά μου και είναι αυτή που αρχικοποιεί τη δομή που κρατά τις πληροφορίες για τα MB του καρέ. Όπως φαίνεται θα πρέπει να βρούμε κάποιο τρόπο να τη βελτιστοποιήσουμε.
- Η τρίτη συνάρτηση sad2 είναι μία από αυτές που θα μου δώσει μια σοβαρή αύξηση στην απόδοση, μιας και αυτή η συνάρτηση είναι που εκτελεί το Sum of absolut differences που είναι η σημαντικότερη συνάρτηση σε εφαρμογές συμπίεσης βίντεο. Εδώ με σωστή εκμετάλλευση των δυνατοτήτων της αρχιτεκτονικής xtensa αναμένω να δω σοβαρές βελτιώσεις.
- Η main είναι η κύρια συνάρτηση του προγράμματος και καταναλώνει τόσους κύκλους λόγω των fread και fwrite, τα οποία θα αφαιρεθούν στη βελτιστοποίηση. Άρα απλά την αγνοούμε.
- Οι συναρτήσεις go_left,go_right είναι συναρτήσεις του αλγορίθμου της συνένωσης και δεν μπορούνε να βελτιστοποιηθούν περαιτέρω στην αρχιτεκτονική xtensa, άρα τις αγνοούμε.
- Η boundary_extension3 είναι η συνάρτηση που αυξάνει τα όρια του καρέ για την εφαρμογή του mvfast και είναι ήδη βέλτιστη με χρήση memcpy.
- Για την merging7_prep, η οποία είναι συνάρτηση αρχικοποιήσεων για την συνένωση θα

βρούμε τρόπο να τη βελτιστοποιήσουμε παρακάτω.

- Τέλος, όπως και για την sad2, έτσι και οι difference2,variance2 οι οποίες υπολογίζουν, η πρώτη τα υπολειπόμενα pixels και η δεύτερη την διασπορά αυτών, έχουν σοβαρές τάσεις για βελτιστοποίηση στην αρχιτεκτονική xtensa και θα δούμε πως θα πετύχουμε την καλύτερη απόδοση γι αυτές παρακάτω.

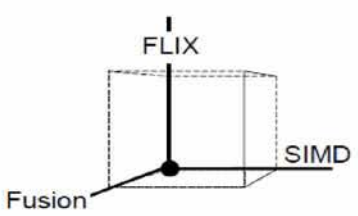
Έτσι το κλειδί για τη βελτιστοποίηση του κώδικα είναι κυρίως οι συναρτήσεις *sad2,difference2 και variance2*, όπου η αρχιτεκτονική έχει φοβερές δυνατότητες με την εφαρμογή simd 128bit, fusion και flix.

Πώς θα βελτιστοποιηθεί ο κώδικας;

Sum of absolut differences

Αφού βρήκαμε τα σημεία τα οποία χρίζουν βελτιστοποίησης, θα χρησιμοποιήσουμε τις τεχνικές της αρχιτεκτονικής Xtensa για να κάνουμε τον κώδικά μας όσο το δυνατόν ταχύτερο.

Όπως τονίσαμε στην αρχή, ο σκοπός μας από τη πρώτη στιγμή, ήταν να δημιουργήσουμε ένα πρόγραμμα που θα τρέχει σε πραγματικό χρόνο, για να το πετύχουμε όμως αυτό, θα πρέπει η υλοποίησή μας να τρέχει σε φορητές συσκευές, οι οποίες έχουν περιορισμένη επεξεργαστική δύναμη σε σχέση με τους σταθερούς υπολογιστές. Η μέση επεξεργαστική δύναμη σήμερα για μία τέτοια συσκευή είναι τα 500 MHz. Επομένως απαιτείται μια υλοποίηση που θα καταναλώνει το πολύ 500 εκατομμύρια κύκλους για κάθε 25 καρέ ή 30 καρέ (αναλόγως αν μιλάμε για PAL ή NTSC αντίστοιχα), εάν θέλουμε να επιτύχουμε το σκοπό μας.



Σχήμα 43:Όλες οι δυνατές τεχνικές βελτιστοποίησης της xtensa, σχηματίζουν ένα τρισδιάστατο χώρο, στον οποίο προσπαθούμε να επιτύχουμε το καλύτερο αποτέλεσμα.

Πρέπει να εκμεταλλευτούμε τις δυνατότητες της xtensa στο έπακρο για να επιτύχουμε αυτό το σκοπό:

- Fusion: Ονομάζει η tensilica τη διαδικασία κατά την οποία, πολλές απλές διεργασίες ενώνονται από τον επεξεργαστή xtensa (xtensa processor) σε μία πολύπλοκη διεργασία. Αυτή η ένωση επιτρέπει στον επεξεργαστή να κάνει περισσότερους υπολογισμούς ανά εντολή, να δημιουργήσει ευκαιρίες διαμοιρασμού εσωτερικών μεταβλητών και να εξαλείψει την ανάγκη για σώσιμο και φόρτωση ενδιάμεσων μεταβλητών.

  Ένα χαρακτηριστικό παράδειγμα είναι το sum of absolut differences στο οποίο αναφερθήκαμε και παραπάνω. Όπως φάνηκε και από την καταγραφή αποτελεσμάτων (profiling) αποτελεί την σημαντικότερη εργασία στον κώδικά μας και είναι το πρώτο που πρέπει να βελτιστοποιήσουμε.

  Μια απλή υλοποίηση της εργασίας sad, σε ένα επεξεργαστή γενικού σκοπού,για 16x16 MB, απαιτεί όπως είναι κατανοητό, 256 φορτώσεις από τη μνήμη,256 προσθέσεις, 256 αφαιρέσεις και 256 υπολογισμούς απόλυτης τιμής. Αυτό σημαίνει 256*4=1024 πράξεις, δηλαδή 1024 κύκλους μηχανής, στην καλύτερη των υλοποιήσεων.

  Με τη τεχνική Fusion εμείς συνθέτουμε μία νέα TIE(Tensilica Instruction Extension) η οποία κάνει 1 φόρτωση από τη μνήμη, 1 πρόσθεση,1 αφαίρεση και 1 υπολογισμό απόλυτης τιμής ταυτόχρονα. Έτσι ο επεξεργαστής την τροποποιεί έτσι ώστε να εκτελείται μέσα σε ένα κύκλο μηχανής του και επομένως το κόστος του υπολογισμού πέφτει στους 256 κύκλους.

  Και η βελτίωσή του είναι της τάξεως του 1024/256=4.
- SIMD: Η αρχιτεκτονική της tensilica, επιτρέπει την υλοποίηση εντολών SIMD(Single Instruction Multiple Data). Οι εντολές αυτές επιτρέπουν την εκτέλεση πράξεων πάνω σε πολλά δεδομένα ταυτόχρονα. Αυτό γίνεται με φόρτωση πολλών τιμών σε μία μεταβλητή μεγάλου εύρους (64 ή 128 bit) και η εκτέλεσή της πράξης σε όλο το μήκος της. Στη περίπτωσή μας ασχολούμαστε με pixel Y(luminance pixels), δηλαδή 8 bit τιμές με εύρος 0-255. Άρα σε μία μεταβλητή 128 bit μπορούμε να φορτώσουμε 16 τέτοιες τιμές και να κάνουμε μία πράξη σε 16 τιμές ταυτόχρονα.

  Αυτό έχει σαν αποτέλεσμα να χρειαζόμαστε πλέον 16 φορτώσεις από τη μνήμη, 16 προσθέσεις,16 αφαιρέσεις και 16 υπολογισμούς απόλυτων τιμών δηλαδή 16*4=64 εντολές, άρα και 64 κύκλους μηχανής.

  Η βελτίωση μου είναι της τάξεως του 1024/64=16, όπως είναι και αναμενόμενο.
- FLIX: Είναι και η νεώτερη τεχνική βελτιστοποίησης της αρχιτεκτονικής, που συναντάται μόνο στους LX πυρήνες(οι πιο νέοι επεξεργαστές της tensilica).

Επιτρέπει την ένωση πολλών εντολών μαζί σε μία, μεγαλύτερου μήκους(32 ή 64 bit), επιτρέποντας έτσι στον επεξεργαστή να εκτελέσει πολλές εντολές ταυτόχρονα με μία εκτέλεση.

Η διαφορά της με την τεχνική fusion είναι πως η fusion συνδυάζει πολλαπλές εργασίες μαζί σε μία εντολή ενώ η τεχνική FLIX επιτρέπει την εκτέλεση πολλαπλών εντολών μέσα σε μία μόνο εντολή, σε ένα κύκλο μηχανής πάντα.

Όταν η FLIX τεχνική συνδυαστεί με την καλύτερη υλοποίηση έως τώρα, την SIMD, τότε θα έχουμε μία εντολή η οποία θα περιέχει μέσα 1 φόρτωση από τη μνήμη, 1 πρόσθεση, 1 αφαίρεση και 1 υπολογισμό απόλυτης τιμής σε 16 τιμές ταυτόχρονα. Πλέον για τον υπολογισμό του sad 16x16 απαιτούνται 16 πράξεις FLIX, άρα μόνο 16 κύκλοι μηχανής.

Έτσι βλέπουμε εύκολα την τεράστια βελτίωση στην απόδοση που πέτυχε η αρχιτεκτονική Tensilica Xtensa στην κυριότερη εργασία του κώδικά μας, αφού επιταχύναμε την εργασία 64 φορές.

Αντίστοιχα επιταχύνονται και οι εργασίες του υπολογισμού υπολειπόμενων pixels (difference) και του υπολογισμού της διασποράς των υπολειπόμενων pixels (variance) με αντίστοιχες βελτιώσεις, χρησιμοποιώντας SIMD και FLIX.

Difference

Η συνάρτηση difference2, είναι και αυτή μια συνάρτηση που μπορεί να βελτιωθεί στην αρχιτεκτονική xtensa.

Αυτό που κάνει, είναι να υπολογίζει τις διαφορές(residuals) ή αλλιώς υπολειπόμενα pixels μεταξύ του MB που βρέθηκε σαν βέλτιστη επιλογή στο καρέ αναφοράς και του MB στο τρέχον καρέ.

Δηλαδή για 16x16 MB, κάνει 512 φορτώσεις, 256 αφαιρέσεις και 256 αποθηκεύσεις στη μνήμη. Αυτό σε x86 αρχιτεκτονική απαιτεί το λιγότερο 1024 κύκλους μηχανής (σε ένα επεξεργαστεί όπου κάθε μία από αυτές καταλαμβάνει ένα κύκλο μηχανής).

Στην αρχιτεκτονική xtensa, μπορούμε να το βελτιστοποιήσουμε, αρχικά με fusion, δημιουργώντας μία εντολή, την sub&save η οποία φέρνει τις δύο τιμές από τη μνήμη υπολογίζει τη

διαφορά και σώζει. Λόγω της μονής μονάδος προσπέλασης μνήμης (load/store unit) εδώ θα χρειαστεί 2 κύκλους για τις φορτώσεις και 1 κύκλο για την αφαίρεση και το σώσιμο στη μνήμη. Άρα 3*256=768 κύκλους, βελτίωση ίση με 25%.

Εφαρμόζοντας SIMD, θα χρειαστεί 2 κύκλους για 16 φορτώσεις ,1 κύκλο για την αφαίρεση 16 τιμών και 1 κύκλο για το σώσιμο 16 τιμών, άρα 64 κύκλους συνολικά.

Τέλος, εφαρμόζοντας FLIX, θα χρειαστεί 48 κύκλους, γιατί το σώσιμο θα γίνει στον ίδιο κύκλο με την αφαίρεση. Αν όμως εισάγουμε και δεύτερη μονάδα προσπέλασης μνήμης(κάτι εφικτό στην αρχιτεκτονική Xtensa) η διαδικασία θα ολοκληρωθεί σε 32 κύκλους συνολικά, μιας και οι δύο αρχικές φορτώσεις μπορούν να γίνουν πλέον παράλληλα.
Άρα στη συγκεκριμένη διαδικασία η xtensa καταφέρνει βελτίωση της απόδοσης 32 φορές.

Variance

Τέλος είναι και ο υπολογισμός της διασποράς των υπολειπόμενων pixels. Για τον υπολογισμό της διασποράς ενός 16x16 MB, χρειαζόμαστε 256 φορτώσεις από τη μνήμη,256 υψώσεις στο τετράγωνο ή πολλαπλασιασμούς, 256 προσθέσεις για το μέσο τετράγωνο και 256 για τον μέσο, ενώ παράλληλα άλλη μία ύψωση στο τετράγωνο , 2 διαιρέσεις, 1 αφαίρεση , 1 ρίζα και 1 σώσιμο. Όλα αυτά μας κάνουν 1030 πράξεις, οι οποίες στην καλύτερη περίπτωση θα εκτελεστούν σε 1030 κύκλους, σε μία αρχιτεκτονική x86.

Με υλοποίηση σε xtensa και χρήση fusion, θέλουμε δύο νέες εντολές, την fetch&add για τον μέσο όρο και την fetch&pow&add για τον μέσο όρο του τετραγώνου, έτσι η διαδικασία θα εκτελεστεί σε 256+256+6=518 κύκλους μηχανής, άρα σχεδόν στο μισό χρόνο.

Με SIMD, θα εφαρμόσω αυτές τις 2 εντολές σε 16 τιμές παράλληλα και θα έχω 16+16+6=38 κύκλοι μηχανής, άρα επιτάχυνση της τάξεως του 27.

Τέλος, με χρήση FLIX αυτές οι δύο εντολές μπορούν να τρέχουν παράλληλα(μιάς και έχουμε 2 μονάδες προσπέλασης μνήμης) και άρα 16+6=22 κύκλοι μηχανής, με επιτάχυνση της τάξεως του 46.

Περαιτέρω βελτιώσεις της αρχιτεκτονικής

Είδαμε πως γράφοντας καίριες διεργασίες του κώδικά μας με βελτιστοποιήσεις της Xtensa επιτυγχάνουμε καλύτερη απόδοση.

Άλλο ένα σοβαρό πρόβλημα όμως στον κώδικά μας, είναι όπως και σε όλες τις εφαρμογές βίντεο, οι φορτώσεις από τη μνήμη και οι αστοχίες της κρυφής μνήμης(cache).

Όταν μία εντολή load ή save εκτελείται, για να χωρέσει σε ένα κύκλο μηχανής, θα πρέπει τα δεδομένα να βρίσκονται στη τοπική μνήμη ή στην κρυφή μνήμη. Σε διαφορετική περίπτωση, η εντολή θα ξαναεκτελεστεί με ελάχιστη καθυστέρηση πέντε κύκλων, εάν το δεδομένο δεν βρίσκεται στη κρυφή μνήμη και επτά κύκλων αν έχουμε αστοχία κρυφής μνήμης. Φυσικά δεν είναι μόνο το κόστος των παραπάνω κύκλων, αλλά και το pipeline hazard (ο κίνδυνος κατά τον οποίο η καθυστέρηση μία εντολής συμπαρασύρει και όσες ακολουθούν στην καθυστέρηση, με αποτέλεσμα να παγώνει όλη η εκτέλεση) που προκύπτει, μιας και η καθυστέρηση θα συμπαρασύρει και όλες τις άλλες εντολές που περιμένουν το αποτέλεσμα της συγκεκριμένης εντολής προσπέλασης μνήμης.

Η πρώτη πιθανή λύση για αύξηση του bandwidth της τοπικής μνήμης είναι η εισαγωγή μιας δεύτερης μονάδας προσπέλασης μνήμης.

Η τεχνική FLIX εισήγαγε την χρήση δύο μονάδων προσπέλασης μνήμης στην αρχιτεκτονική xtensa, μια εφαρμογή που θα βελτιώσει ακόμη περισσότερο τον κώδικά μας σε επίπεδο προσπέλασης μνήμης, καθώς, πλέον σε μία εντολή FLIX μπορεί και να διαβάζει και να γράφει ο επεξεργαστής ταυτόχρονα. Έτσι η καινοτομία αυτή, μαζί με την προσπέλαση σε μεταβλητές simd 128 bit μας δίνει το θεωρητικό bandwidth των 11.2 GB/s , για ένα επεξεργαστή 350 Mhz.

Ακόμη και τώρα όμως, το παραπάνω bandwidth που αναφέρθηκε μπορεί να μην είναι εφικτό, μιας και τα δεδομένα μπορεί να μην βρίσκονται στην τοπική μνήμη και να πρέπει να έρθουν από κάποια εξωτερική πηγή. Παράλληλα υπάρχει και το ενδεχόμενο να μην χωράνε στην τοπική μνήμη οι δομές δεδομένων του κώδικα. Εδώ θα χρειαστούμε κάτι πιο δραστικό.

Το inbound PIF της αρχιτεκτονικής xtensa είναι μια μονάδα η οποία μεταφέρει δεδομένα από και προς τη τοπική μνήμη του επεξεργαστή χωρίς να χρησιμοποιεί εντολές προσπέλασης μνήμης. Αντίθετα, επιτρέπει σε μια μονάδα να μεταφέρει δεδομένα μέσα και έξω από τη τοπική μνήμη όταν ο επεξεργαστής δεν προσπελαύνει αυτά τα δεδομένα.

Μία καλύτερη υλοποίηση, για μέγιστο bandwidth μεταφορών του inbound PIF, είναι να εξοπλίσουμε τον επεξεργαστή με δύο τοπικές μνήμες δεδομένων και να δώσουμε στο PIF την μεγαλύτερη προτεραιότητα. Έτσι το bandwidth μεγιστοποιείται, όσο οι load/store εντολές γράφουν

και διαβάζουν στη μία τοπική μνήμη, ενώ η άλλη χρησιμοποιείται για το inbound PIF.

Έτσι έχουμε ένα μηχανισμό ο οποίος βάζει πάντα στην τοπική μνήμη ότι δεδομένο είναι χρήσιμο για τον αλγόριθμο και με αυτό τον τρόπο μειώνονται οι αστοχίες κρυφής μνήμης.

Εφαρμόζοντας όλες τις παραπάνω βελτιστοποιήσεις, με SIMD 128 bit, TIE modules/fusion και FLIX εντολές 64 bit, καθώς επίσης και με 2 μονάδες προσπέλασης μνήμης, από τα οποία το πρώτο ασχολείται με τις εντολές προσπέλασης μνήμης, ενώ το δεύτερο χρησιμοποιείται για τις μεταφορές μνήμης στο inbound PIF έχουμε μειωμένες αστοχίες κρυφής μνήμης και επιτυγχάνουμε ένα αποτέλεσμα για 25 καρέ βίντεο σε 300 εκατομμύρια κύκλους μηχανής στην αρχιτεκτονική Tensilica xtensa. Το αποτέλεσμα αυτό είναι πολύ κάτω σε σχέση με τους 500 εκατ. κύκλους που είχαμε θέση σαν όριο και άρα μόνο καλό μπορεί να θεωρηθεί.

Console | Profile (Cycles) | Call-Graph | Comparison | Saved Output | Pipeline | ISA Profile

Function Name	Total cycles (...	Function cycles	Children cycles	Total cycles	Times called (...	Size (byte
<TOTAL>	100,00	0	302.288.414	302.288.414	0	7.9
main	35,60	107.619.721	194.666.256	302.285.977	1	2.0
sad2	27,16	82.124.131	647.955	82.772.086	522.078	4
memset	11,41	34.518.136	0	34.518.136	110.437	
memcpy	8,99	27.190.503	990	27.191.493	46.914	2
difference2	4,48	13.546.656	0	13.546.656	39.150	2
merging5	2,18	6.614.027	0	6.614.027	29	
small_diam	1,75	5.299.640	38.624.231	43.923.871	29.162	3

Σχήμα 44:Το profiling στην αρχιτεκτονική xtensa μετά την εφαρμογή των βελτιστοποιήσεων.

Το σημαντικότερο όμως είναι, πως σύμφωνα με το αποτέλεσμα μας, ο κώδικάς μας μπορεί να τρέξει σε ένα επεξεργαστή tensilica xtensa με την αρχιτεκτονική που δημιούργησε ο μεταγλωττιστής της tensilica σε πραγματικό χρόνο, εφόσον αυτός είναι 300MHz και πάνω.

Κεφάλαιο 10ο:Συμπεράσματα

Εξήγηση και αξιολόγηση αποτελεσμάτων

Αυτό που καταφέραμε με την υλοποίησή μας, είναι ένα πρόγραμμα το οποίο αναγνωρίζει τα κινούμενα αντικείμενα ενός ικανοποιητικού σε ποιότητα βίντεο. Και λέμε ικανοποιητικού γιατί όπως δείξαμε παραπάνω ο κώδικας μειώνει την αποτελεσματικότητά του όσο χαμηλώνει η ποιότητα του βίντεο, όσο κινείται η συσκευή εγγραφής και όσο δυσανάλογη είναι η κίνηση των αντικειμένων (πολύ αργή/πολύ γρήγορη/περιστροφή).

Σε κάθε άλλη περίπτωση, βγάζει πολύ καλά αποτελέσματα, τα οποία μπορούν να έχουν σοβαρές εφαρμογές, μιας και γνωρίζοντας τις κινήσεις των αντικειμένων μπορούμε να βρούμε και την ταχύτητα με την οποία αυτά κινούνται, εφόσον γνωρίζουμε την απόσταση τους από την πηγή λήψης. Ταυτόχρονα παίρνουμε πληροφορία για το κάθε αντικείμενο, σχετικά με το αν κινείται, προς τα που και με τι ταχύτητα, καθώς επίσης και τι σχήμα έχει.

Άλλες εφαρμογές που μπορεί να έχει, είναι στην επεξεργασία εικόνας, μιας και σε εικόνες υπάρχει απαίτηση να αναγνωρισθεί το φόντο από μία εικόνα, για να γίνουν σε αυτό κάποιες απαιτούμενες

αλλαγές(χρωματισμός,αλλαγή κ.λ.π).

Το δυνατότερο σημείο του όμως είναι, πώς με τη χρήση του αλγορίθμου mvfast για εκτίμηση κίνησης αλλά και με βελτιστοποιημένες μεθόδους για υπολογισμό του Sum of absolut differences, για τις διαφορές(residuals-differences), αλλά και για τη διασπορά(variance), καταφέραμε να εκμεταλλευτούμε μία νέα σχετικά αρχιτεκτονική η οποία μπορεί να έχει πολλές εφαρμογές στο μέλλον για να τρέξουμε σε πραγματικό χρόνο την εφαρμογή μας σε φορητές συσκευές, οι οποίες χρησιμοποιούνται ευρέως πλέον από τον κόσμο.

Έτσι ένα άτομο θα μπορέσει, κρατώντας το κινητό του τηλέφωνο στο χέρι του, να έχει ανοιχτεί την κάμερα του και όσα καταγράφει η κάμερα εκείνη τη στιγμή να επεξεργάζονται από την υλοποίηση μας και να βλέπει κατευθείαν στην οθόνη του όλα τα κινούμενα αντικείμενα και όλη την πληροφορία που αναφέραμε σχετικά με αυτά.

Αλλά και με εφαρμογή σε έξυπνα φανάρια θα μπορούμε να παίρνουμε πληροφορία σχετικά με ένα πιθανό τρακάρισμα, όταν η κάμερα παρατηρεί δύο αντικείμενα με διαφορετικά διανύσματα κίνησης, να συμπέσουν και στη συνέχεια να αποκτούν ένα κοινό διάνυσμα κίνησης, δηλαδή να γίνονται ένα αντικείμενο.

Ακόμη, κάμερες υψηλής ευαισθησίας και εμβέλειας, μέσα σε μεγάλες δασικές εκτάσεις, θα σαρώνουν την επιφάνεια του δάσους, πάνω από τα δέντρα για αναγνώριση καπνού και θα ειδοποιούν την πυροσβεστική υπηρεσία για πιθανή πυρκαγιά στην περιοχή.

Τέλος, κάμερες ασφαλείας, θα αναγνωρίζουν μέσω ενός δοθέντος σχήματος (default pattern) την μορφή ενός ανθρώπου (πιθανόν ληστή) σε κάποιο χώρο ασφαλείας και θα ειδοποιούν για ενδεχόμενο κλοπής, αλλά και γενικά για παρατήρηση κίνησης μέσα σε κάποιο χώρο.

Έτσι αξιολογώντας το αποτέλεσμα της υλοποίησής μας, αναγνωρίζουμε πως το ισχυρότατο πλεονέκτημά της είναι η εφαρμογή σε πραγματικό χρόνο.

Θέματα που επιφέρουν βελτίωσης

Όπως θα έχει γίνει ήδη κατανοητό έως τώρα, το σημαντικό μειονέκτημα της υλοποίησής μας, είναι πως δεν δουλεύει για χαμηλής ποιότητας βίντεο και ανομοιογενή κίνηση των αντικειμένων.

Αντοχή στο θόρυβο

Όταν ένα βίντεο είναι χαμηλής ποιότητας, παρατηρούμε διάφορα σημεία, τα οποία αν και δεν υπάρχει κίνηση (π.χ. Ουρανός, ή δρόμος) κάνουν κάποια “σπασίματα” δηλαδή απότομες αλλαγές σε χρώμα. Αυτό δυστυχώς δίνει λανθασμένη εικόνα ενός κινούμενου αντικειμένου στην υλοποίησή μας και όταν πρόκειται για κάτι μικρό, τότε μπορούνε να αγνοηθούν από τον κώδικα μέσω του κατωφλίου που είχαμε θέση παραπάνω, για μεγαλύτερο πρόβλημα όμως κάτι τέτοιο δεν είναι εφικτό οπότε δεν μπορεί να υπάρξει κάποια μελλοντική βελτίωση σε αυτό τον τομέα.

Ταυτόχρονα, τα βίντεο χαμηλής ποιότητας, τείνουν να διασπείρουν την κίνηση και στα γύρω pixel από αυτά που κινούνται, λόγω θορύβου, και έτσι δίνει στον κώδικά μας μια λανθασμένη ένδειξη κίνησης και σε γύρω pixel που κανονικά είναι ακίνητα.

Έτσι, σε θέματα θορύβου, η υλοποίησή μας είναι καταδικασμένη να αποτυγχάνει.

Περιστροφή,μεγέθυνση/σμίκρυνση αντικειμένου

Η περιστροφή ενός αντικειμένου, δεν μπορεί να αναγνωριστεί από τον κώδικά μας, και αυτό γιατί τα σημεία στις άκρες κινούνται, ενώ αυτά κοντά στον άξονα περιστροφής όχι, έτσι η κίνηση δεν είναι ομοιογενής και ο αλγόριθμός μας αποτυγχάνει να βρει σωστά το αντικείμενο.

Αντίστοιχα και στην μεγέθυνση αλλά και στη σμίκρυνση , έχουμε το ίδιο πρόβλημα, κάποια pixel του αντικειμένου κινούνται, κάποια άλλα λιγότερο και κάποια άλλα καθόλου. Άρα δεν έχουμε ομοιογενή κίνηση, άρα ο αλγόριθμος αποτυγχάνει.

Κίνηση της συσκευής λήψης

Αναγνωρίσαμε πιο πάνω, πως άλλο ένα πρόβλημα της υλοποίησής μας, είναι πως δεν καταφέρνει να βρει τα αντικείμενα αν η κάμερα κινείται. Αυτό συμβαίνει, γιατί όλο το καρέ κινείται με κάποια ταχύτητα και έτσι αδυνατεί να βρει φόντο.

Σε αυτή την περίπτωση, μία λογική λύση θα ήταν ο κώδικας μας βρίσκει το μέσο διάνυσμα

κίνησης όλων των MB και να το θέτει ως την κίνηση του φόντο, έτσι θα μπορεί να υπολογίσει όλες τις σχετικές κινήσεις των αντικειμένων που κινούνται και να βγάλει ένα ασφαλές συμπέρασμα για το τι κινείται ή όχι.

Επομένως, εδώ φαίνεται ότι υπάρχει μία σοβαρή υποψηφιότητα για μελλοντική βελτίωση, μιας και με αυτό τον τρόπο η υλοποίηση μας θα καταστεί εφικτό να αντιμετωπίζει καρέ που κινούνται ολόκληρα.

Περαιτέρω βελτίωση της ταχύτητας του κώδικα

Ακόμη και μετά την εφαρμογή του inbound PIF, παρατηρούμε ότι το πρόβλημα των αστοχιών μνήμης(cache misses) εξακολουθεί να χρήζει βελτίωσης.

Μία λογική βελτίωση, είναι η σχεδίαση ενός επεξεργαστή, ο οποίος διαθέτει τοπική μνήμη, τόση, όση χρειάζεται για να αποθηκευτούν δύο 16x16 MB, δηλαδή 256*2=512 bytes, υποθέτοντας ότι στον επεξεργαστή μας ο unsigned char καταλαμβάνει 1 byte.

Με αυτό τον τρόπο, θα μπορέσω να αποθηκεύω μέσα της τα δύο macroblocks που κάθε φορά πρόκειται να χρησιμοποιήσω για να βρω το sad και να τα φέρω από τη μνήμη κάθε φορά στην αρχή της συνάρτησης που υπολογίζει τα sad.

Μπορώ να κάνω το ίδιο για τα απαραίτητα pixels των 2 καρέ, κατά την αρχή της συνάρτηση difference, και για τα δεδομένα (residuals) που θα χρειαστώ για τον υπολογισμό της διασποράς(variance) και έτσι , θα γλιτώσω τις περισσότερες από τις αστοχίες κρυφής μνήμης που συναντώ .

Παράλληλα, όπως είδαμε από το profiling, αρκετές συναρτήσεις αρχικοποιήσεων αλλά και η boundary extension3η οποία επεκτείνει το καρέ προς όλες τις κατευθύνσεις για την εφαρμογή του αλγόριθμου για εκτίμηση κίνησης, καταναλώνουν πολλούς κύκλους, κάτι που αν μεταφερθεί στην αρχιτεκτονική xtensa, θα εξοικονομηθούν αρκετοί κύκλοι ακόμη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- *Performance report of Motion Vector Field Adaptive Search Technique by Kai-Kuang Ma and Prabhudev Irappa Hosur
- *Fast motion estimation within the H.264 codec by Alexis Michael Tourapis
- *Predictive Motion Vector Field Adaptive Search Technique (PMVFAST)-Enhancing Block Based Motion Estimation by Alexis Tourapis,Oscar Au and Ming Liou.
- Tensilica Training Slides
- Tensilica Xtensa Instruction Set Architecture-Reference Manual
- Tensilica Instruction Extension(TIE) Language-User's Guide
- Tensilica Instruction Extension(TIE) Language-Reference Manual
- Digital Image Processing by Rafael Gonzalez
- Video Codec Design by Iain Richardson
- Τεχνολογία Πολυμέσων –Θεωρία και Πράξη από τους Σ.Ν. Δημητριάδης, Α.Σ. Πομπόρτσης και Δ.Γ. Τριανταφύλλου
- High-Fidelity Multichannel Audio Coding by Dai Tracy Yang, Chris Kyriakakis and C.-C. Jay Kuo
- *Οι αναφορές με αστερίσκο (*) αποτελούνε άρθρα*

gebardas@uth.gr

ΕΥΧΑΡΙΣΤΙΕΣ

- Την οικογένεια μου για την οικονομική και ψυχολογική βοήθεια
- Τους επιβλέποντες καθηγητές και συγκεκριμένα τον Γ. Κατσαβουνίδη, για τη συνεχή καθοδήγηση κατά τη διάρκεια της εργασίας μου και τη συνεργασία μας, η οποία ήταν άψογη σε όλη της τη διάρκεια. Αλλά και για την πρόταση του θέματος της διπλωματικής μου εργασίας.
- Τους φίλους, για την καθημερινή υποστήριξη
- Τον Στέργιο Πουλαράκη, για την βοήθεια και υπομονή του
- Τον Χρήστο Αλιμήση, για την διαρκή ενημέρωση
- Και όλους τους υπόλοιπους που συνεισφέρανε για να ολοκληρωθεί αυτή η εργασία

gebardas@uth.gr